RICE UNIVERSITY

Unsynchronized Distributed Motion Planning with Safety Guarantees under Second-Order Dynamics

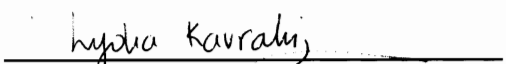
by

Devin Kieber Grady

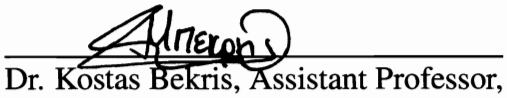
A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

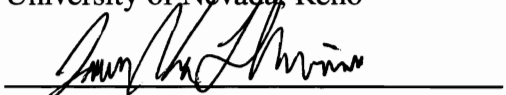
APPROVED, THESIS COMMITTEE:



Dr. Lydia E. Kavraki, Chair, Noah Harding Professor, Computer Science



Dr. Kostas Bekris, Assistant Professor, Computer Science and Engineering, University of Nevada, Reno



Dr. James McLurkin, Assistant Professor, Computer Science

Dr. Mark Moll, Research Scientist, Computer Science

Dr. Marcia K. O'Malley, Associate Professor, Mechanical Engineering

HOUSTON, TEXAS

APRIL, 2011

ABSTRACT

Unsynchronized Distributed Motion Planning with

Safety Guarantees under Second-Order Dynamics

by

Devin Kieber Grady

Robots are increasingly found to operate together in the same environment where they must coordinate their motion. Such an operation is simple if the motion is quasi-static. Under second-order dynamics, the problem becomes challenging even for a known environment. Planning must guarantee safety by ensuring collision-free paths for the considered period by not bringing the robot to states where collisions are inevitable. This can be addressed with communication among robots, but it becomes complicated when the replanning cycles of different robots are not synchronized and robots make planning decisions at different times. This thesis shows how to guarantee safety for communicating second-order vehicles, whose replanning rates do not coincide, through a distributed motion planning framework without a global time reference. The method is evaluated through simulation where each robot has its own address space, and communicates with message passing. A proof of safety is presented, and simulation results are used to investigate performance of the framework.

Acknowledgements

I would like to thank my adviser, Dr. Lydia Kavraki, for her guidance and dedication to helping me produce quality research. This work would also not be possible without the prior efforts of Dr. Kostas Bekris, who laid the foundations for this work and was instrumental in developing the theoretical underpinnings of my research. My thesis committee deserves thanks for spending their valuable time on my behalf.

Additionally, the entire Physical and Biological Computing Group at Rice for providing a comfortable, innovative atmosphere to work in, and for their valuable criticism.

Finally, I would like to acknowledge my friends and family for their support and understanding.

This work was funded in part by Rice University, the US Army Research Laboratory and the US Army Research Office under grant number W911NF-09-1-0383, NSF IIS 0713623, and through an NSF Graduate Research Fellowship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis considers multiple autonomous robots with complex dynamics operating in a static environment. The robots try to reach their individual goals without collision. Such scenarios are becoming increasingly interesting. For instance, consider the case of vehicles moving in a parking lot or going through a busy intersection, or unmanned aerial vehicles that carry out sophisticated maneuvers. These examples involve second-order systems, which cannot stop instantaneously and must respect limits in the second-order derivatives of their state parameters. For these systems, such as a car, guaranteeing collision avoidance with obstacles in the environment cannot be easily accomplished. If these obstacles are other moving robots, the problem becomes even harder. Reasoning about the future actions of these robots must be done to avoid collision. There exist many real applications that require the solution of such problems in a decentralized manner because the problem difficulty grows with the number of robots otherwise.

Many of these applications arise every day. A robotic construction site is a particu-

larly motivating example. Even "minor" collisions among vehicles carrying heavy loads could have disastrous consequences because of their momentum. Not only is it meaningful to hope for robotic construction to be safer and more productive than human construction sites, robotic applications are not limited to human-viable environments. It is not unreasonable to hope that humans will be exploring new planets, using pre-built structures set up by the robots sent on ahead.

Similar to construction, robotic mining has major potential to be safer and more productive. Currently, large tunnels need to be constructed for ventilation and just so that people can fit. Robot workers could be designed to be different shapes, and cut tunnels sized around the vein of mineral. This would not only save refining effort and be safer, but could possibly cut down on some of the environmentally damaging aspects of mining.

These applications need a robotic framework that is capable of dealing with different types of robots easily, so that the team can be designed around the task. A specialized application team, requiring reprogramming with each change in membership, would not scale well in the task domain. Therefore, we desire a planning scheme that can be used to coordinate between different robots. Ideally, this scheme would allow the robots to have parameter variation between them, such as exact clock value synchronization. Expecting robots to have identical clock values, when they are turned on and activated in completely separated areas of the environment is unreasonable, particularly if the robots are out of communication range. They are expected to have fairly close agreement on the clock rates, however. Most modern electronics provide stable clock signals, so this is not expected to be a challenge, and is assumed to be available. Finally, it is desirable to not place a strict upper limit on the team size, so that jobs of any size and complexity can eventually be ad-

dressed. As such, the scheme should follow a decentralized approach to avoid exponential computational requirements.

However, the general mover problem, defined as finding a path for a linked polyhedral object moving in a polyhedral environment to a specific goal configuration has been proven to be PSPACE-Hard [1]. Attempting to solve this problem exactly would be impossible given current algorithmic techniques, and the multi-robot coordination task is significantly more complex. The dimensionality of the problem increases with every additional robot. This helps explain why it is key that we develop a decentralized solution to avoid the exponential increase in computation required. The price that we pay is giving up on the completeness of our approach, so we might not be able to solve particular problems that do admit a solution. However, it also makes sense for each robot to be as self-sufficient as possible when we envision them in dangerous environments. To meet these desires, this thesis imposes a requirement for a decentralized solution and considers robots that replan their trajectories on the fly.

A decentralized approach is desired to improve the system's ability to scale to larger robotic teams. To coordinate the robots, this work utilizes direct peer-to-peer communication. A planning algorithm makes use of information collected through communication to avoid collisions for the next cycle and ensure that robots reach states from where collisions can be avoided in the future. The duration of the planning cycle is the same for all robots, and they count time from zero at the same rate but the robots do not agree on when time was zero. Therefore, communication of plans can happen at any point and the robots need to operate safely in the presence of partial information about the plans of their neighbors. An unsynchronized, decentralized framework is developed that guarantees the safety of all

robots in this setup.

Replanning is asking the robots to find many small, partial solutions that, when concatenated, end up solving the global problem. Although not complete, it is not surprising given the known computational complexities the problem entails. Replanning also allows robots to consider multiple alternative trajectories during each replanning cycle and provides flexibility in changing environments. A final benefit of planning is that it allows for new robot dynamics to be easily added to the system's capabilities.

Overall, the task is to determine a set of acceptable trade-offs to design a safe algorithm that solves the multi-robot navigation problem. The particular benefits of our approach are:

- provably safe operation,
- generality to a very large class of robot dynamics,
- planning into the future to avoid local minima and deadlock problems,
- explicit communication so that reasoning about the future behavior of robots is simple,
- and it uses a decentralized approach that can scale to large numbers of cooperating vehicles.

However, we accept these tradeoffs:

- a lack of any completeness or liveness guarantee,
- no sensing model requires communication to be guaranteed within a particular range,

- use of decentralized computation means that no robot has knowledge of the global state and therefore cannot easily make decisions that benefit global convergence to a solution.

The complexity of the problem requires selecting appropriate characteristics such that our algorithm can be computed in real-time for robotic applications, but is not so simplified or specialized that it looses its applicability to current robotic teams. This thesis proposes that the above list is a reasonable and acceptable set of trade-offs that allow us to create a coordination protocol that is computationally tractable as well as useful to real systems.

This thesis is a significant departure from previous work because it specifically considers unsynchronized robots. In this case, the robots cannot expect messages arrive at a particular time. This is a significant challenge, however, safety can still be guaranteed. A proof and implementation of a novel, unsynchronized algorithm is presented.

Throughout this thesis, the term unsynchronized will be used to specify that a global time reference frame is not available to any agents. Thus all agents must proceed without any knowledge of specific time information of their neighbors, and only use current state and duration information, combined with the explicit communication proposed by our safety protocol. We do require that expected messages will arrive within a specific bounded time, although we do not impose restrictions on exactly when.

1.1 Contributions

The thesis presents a general framework with reduced assumptions relative to the literature for independent but communicating second-order robots to reach their destinations

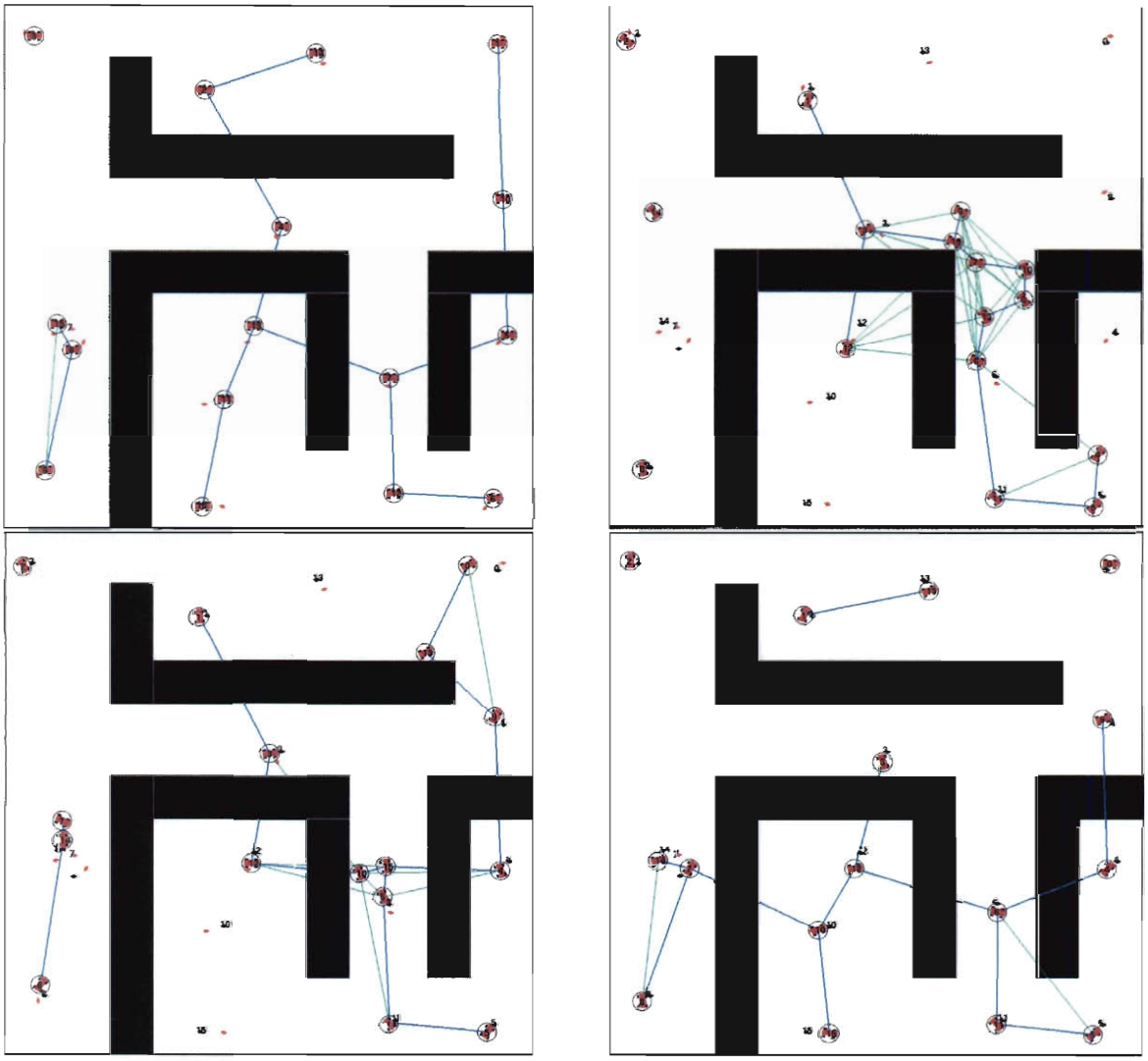


Figure 1.1: A sample run of the decentralized framework presented in this thesis, operating in the office environment (left to right). Links show communicating robots.

in an otherwise known environment. The framework is fully distributed and relies on unsynchronized interaction among the robots, where the replanning cycles of the robots are not synchronized and the robots have no knowledge about their clock differences or access to a global clock. The main contribution over previous work [2] is the relaxation of the

synchronized operation assumption. The challenges that unsynchronized operation pose are diverse, due to unknown message delays and the fact that the planning cycles are no longer all performing the same operations at the same time. It is also no longer possible to determine when any robot will enter a particular state that is has communicated to its neighbors, just that it will at some point in the future. This thesis shows that, although some of the algorithms used in previous work are no longer usable in the unsynchronized framework, absolute safety guarantees can still be made under reasonable assumptions.

Safety guarantees are based on the exchange of contingency plans between neighboring robots that are sure to be collision-free. While contingency plans have been used in the past to provide safety for individual agents [3, 4], this line of research emphasizes the importance in communicating these plans in multi-robot scenarios, and studies specifically the unsynchronized case. A proof that shows that the proposed scheme guarantees collision avoidance is provided. The proof is inductive, so from a safe start state, the system is provably safe for all future time. The framework has been implemented on a distributed simulator, where each robot is assigned to a different processor and message passing is used to convey plans. The experiments consider various scenarios involving 2 to 48 robots and demonstrate that safety is indeed achieved in scenarios where collisions are frequent if the collisions at a future time step that cannot be avoided due to the complex system dynamics are ignored. The experiments also evaluate the efficiency and the scalability of the approach. Although no completeness guarantees can be made for this protocol, we did not have any experiments that were not eventually solved. Path quality also has no guarantees and can be arbitrarily poor, depending on the metric used. This largely depends on the particular planning solution used as a component in this coordination scheme, as

well as the particular problem that is posed and the specific robot dynamics.

Chapter 2

Background

The multi-agent navigation problem for robots with second order dynamics is multi-faceted. The problem is, informally, to get n robots from configurations $A_0 \cdots A_n$ to configurations $B_0 \cdots B_n$ without crashing into each other. The main characteristics of this problem are:

1. Navigation - how to get a single robot from a start configuration to a goal configuration.

2. Safety - how to ensure that the paths constructed are safe from collisions.

3. Coordination - how to have multiple robots performing safe navigation tasks in the same shared environment.

2.1 Navigation

The problem of navigation is one of getting a robot from a starting initial position, to a final goal position. More properly, the positions here are really robot configurations. A configuration is a set of parameters that specifies the entire robot state. For a car operating in a plane, this would be position on the plane, orientation on the plane, and forward velocity. Compared to cars, navigation is easier in the case of robots that are holonomic, or move in any direction easily, using e.g., the Global Dynamic Window approach [5]. Holonomic systems have been specially designed [6], and navigation of these robots has been addressed [7]. However, many systems currently deployed are non-holonomic and require more complex navigation solutions. The classic parallel-parking problem, for instance, cannot be solved by these approaches because a car is non-holonomic [8]. Several important classes of non-holonomic systems have been studied, from underactuated joints with dynamics [9, 10] to differential drive, Dubins' car [11], Reeds and Shepp cars [12] and cars with trailers. An overview of these systems was published in 1998 [13] and discusses many important characteristics of each, and expresses that the nonholonomic planning problem is challenging even in an obstacle-free environment. Second-order vehicles are those that have only second derivatives of state variables as control inputs. For example, a car operating on the plane, has state its state defined by position in X and Y, orientation in θ, and first order variables forward velocity and steering angle. The actual controls of this car, however, are second order properties: acceleration and rate of steering change. Navigation for this type of system, as with most robots, can be addressed by either:

- a control law [14, 15] which shows how to produce stable controllers that can guide

a car to the goal,

- or by planning [16, 17, 18], where the controls of the system in this case, acceleration and steering are directly simulated to find a path to the goal configuration.

The primary difference between these two approaches is that a reactive control law, once implemented, is generally expressed by sets of equations that relate the required control inputs to the current state and the desired state. Planning, on the other hand, takes the equations of motion, coupled with the available control inputs, and finds a sequence of inputs that is simulated or solved for some time in the future to discover the inputs that can achieve the desired state. As such, planning tends to be more general across systems with different dynamics and can solve problems in complex environments with closer to globally optimal behavior, although harder to analyze and make guarantees about [19]. However, control theoretic approaches react quickly to any error in state estimation because the new controls are available almost instantly, and therefore can be more robust to deviations in robot state or execution noise.

2.2 Safety

Safety issues for dynamical systems were first studied as early as 1985 [20]. Collision-free states that inevitably lead to collisions have been referred to as Obstacle Shadows [20], Regions of Inevitable Collision [21] or Inevitable Collision States (ICS) [22]. A study on ICS resulted in conservative approximations [22], integration with replanning schemes [4] and generic ICS checkers [23]. It also provided necessary criteria for motion safety under any navigation scheme [24]: a robot must

i) consider its dynamics,

ii) model the environment's future behavior, and

iii) reason over an infinite-time horizon.

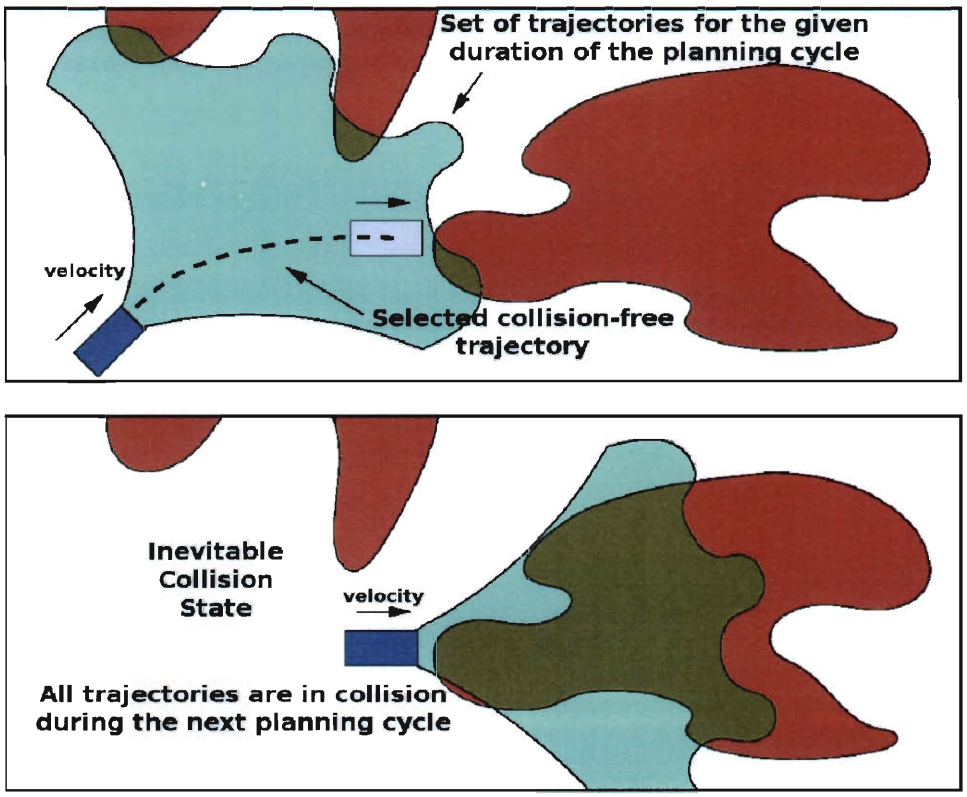


Figure 2.1: Illustration showing an Inevitable Collision State ICS [25]

Earlier work on this problem includes reactive methods. Reactive methods can enable a robot to avoid collisions for unknown or dynamic environments. An early, successful reactive method was the Vector Field Histogram approach [26], which, however, did not reason about robot dynamics. Path deformation methods, where a path is found to the

goal and then reactively deformed to avoid obstacles in the environment enjoy success for robots that can be assumed to be able to track a smooth path in the workspace [27, 28] and iterative path deformation that explicitly respects the nonholonomic constraints of the robot be perturbing the input functions of a reactive controller to find a set of possible paths [29, 30]. However, these methods cannot make guarantees about safety because although they attempt to react to changes in the environment, they do not reason into the future and address the ICS issue. That is, each change to the environment is considered static, although they can update very quickly to changes that occur over time. Although they work well in practice for some robotic systems, they cannot guarantee safety. More recent alternatives do reason about dynamics and include the Nearness Diagram Navigation [31], the Dynamic Window Approach [32, 5, 33] and Velocity Obstacles [34, 35]. The first two assume static obstacles, while the later typically assumes obstacles with constant linear velocity. The above approaches do not provide ICS avoidance. As an example, the Nearness Diagram Navigation and the Dynamic Window approaches do not reason about the motion of obstacles. Many of the popular reactive planners do not satisfy the criteria for motion safety [24] and experimental comparisons have also shown the practical importance of reasoning about ICS in reactive navigation [23].

The work on Reciprocal Velocity Obstacles (RVOs) [36] involves multiple agents which simultaneously avoid one another. RVOs can be used to simulate thousands of moving agents without collisions and achieve this objective without communication but do not deal yet with ICS. These Velocity Obstacles have been recently extended into Acceleration Velocity Obstacles (AVOs) [37], and thus can handle selected second-order systems, however, the three criteria for proven motion safety are not addressed in general and thus they cannot

provide ICS avoidance guarantees. In particular, their collision avoidance scheme provides a parameter τ for the look-ahead, and cannot generally use $\tau = \infty$ without resulting in deadlock. Our proposed system can achieve $\tau = \infty$ safety guarantees through the use of contingencies that are valid for infinite time into the future and therefore can guarantee ICS avoidance, and, although liveness is not guaranteed, our framework has not been experimentally observed to enter deadlock. The robotic systems that they can address are also limited to those where the relative motion of two robots can be determined from the state of the robot and the difference between their control inputs. A related control-based, reactive method [38] is able to deal with second-order models of a planar unicycle but does not provide guarantees in environments with obstacles. Another reactive method provides guarantees about safety and liveness [39], but only for a particular first-order vehicle in an obstacle-free environment.

In this thesis, and in contrast with the above reactive approaches, the focus is on *planning* directly safe paths. Planning reasons about a longer time horizon than reactive methods, which only user the current state, so it does not get stuck in local minima as easily and extends to higher degrees-of-freedom (DOF) systems. Additionally, the planners need to be changed very little to be applied to systems with new dynamics, in stark contrast to writing new control laws. Reasoning about safety during the planning process allows a planner to focus on the safe part of the state space. In this work, planning and replannning with second-order dynamics is achieved using a sampling-based tree planner [21, 3, 25]. Alternatives for the planning process could include, among others, navigation functions [40] and lattice-based approaches [41]. There is no restriction on all agents to use the same planning system, as long as they follow the safety protocol as described.

The literature on replanning in static environments shows that braking maneuvers are sufficient to provide safety and were used within a control-based scheme [42] and in sampling-based replanning [43, 25]. In the case of dynamic environments most of the existing work considers a relaxation of ICS. For instance, the notion of τ-safety was used in a real-time kinodynamic planner for dynamic environments [44]. The τ-safety notion guarantees safety from collisions for τ seconds in the future for each node of a sampling-based tree, but violates the third criterion of motion safety (i.e., reason over an infinite-time horizon). A kinodynamic sampling-based planner was tested on real air-cushioned robots moving in dynamic environments, where an escape maneuver was computed when the planner failed to find a complete trajectory to the goal [3]. Learning-based approximations of an ICS set can also be found [45], as well as approximations for computing state×time space obstacles [46]. Other works focus on the interaction between planning and the sensing limitations of a robot, and point out that it is necessary to limit planning within the robot's visibility region, since the visibility boundary may be hiding moving obstacles [47, 48]. Adaptive replanning cycle times were recently shown [49] to improve consistency of results and safety for replanning systems. Additionally, it was shown that in the case of a deterministic environment and static objective, it provides asymptotic completeness. This work uses a deterministic environment and static objectives, so it is possible that future research will involve the integration of this adaptive replanning cycle time method with the safety protocol presented here.

2.3 Coordination

So far, we have not discussed how to actually coordinate vehicles. The reactive methods often use sensing to obviate the need for explicit communication. This way, the control of robots is actually based on the current state, the goal state, and the states of neighboring robots. All of the approaches discussed above assume this model, or simply assume that the states of all robots are available globally. This thesis deviates from these models by assuming that the robots can be localized accurately on a static map of the environment but does not have a sensor model of any kind. By operating entirely without sensing, we must address coordination through explicit communication of intention and future states. This thesis contributes a coordination protocol that is robust to operating in an unsynchronized fashion and is provably safe in the absence of sensing. The assumptions on perfect localization and communication are reasonable given this model, because it is clear that without either one of these, we could not guarantee safety. To show this is simple: imagine a robot headed towards a stationary robot. If all communication is dropped between the two robots, then, without sensing, they are invisible to each other and the moving robot has no reason to even try to stop before colliding with the stationary robot. A system with bounded unreliable communication can be modeled as perfect communication if, within a certain range, the communication is perfect. Similarly bounded error on localization can be addressed by simply requiring the robots to be farther away from each other at all times. As such, we do not explicitly consider either problem, although in any physical implementation it would need to be addressed in some fashion. The benefit of this approach is that we do not assume any particular sensor model: any set of sensors that can give us reasonable localization can

be used.

With communication added, the motion planning problem can still be broken down and considered as coupled but distributed [50, 51], where robots work together on a collective solution. It could also be completely decentralized [52, 53], where each robot is collaborating on a decision but each is making its own decisions locally. As long as the reliable communication radius is larger than the safe stopping distance, we can use a decentralized approach to ensure safety. In section 3.7, a formula relating communication range and maximum allowed velocity is given, assuming a braking contingency. There has been enough research in these areas to see several overview and comparison papers [54, 55, 56] written on this topic.

Planning dynamic networks of robots has been also approached by a combination of centralized and decoupled multi-robot planners [57], without considering second-order systems or the ICS challenge. In general, multi-robot planning can be approached either with centralized planning, which is typically not scalable but is theoretically complete and will always find a solution if it exists, or decoupled approaches, which may involve prioritization of robots [58] or velocity tuning [59] and are not complete. The main difference between these approaches is that solving the coupled problem involves reasoning about all robots' control inputs at once, while in the decentralized method, only the local robot and (perhaps) its neighborhood are reasoned about. The computational costs rise exponentially with the number of robots considered in the centralized case [60]. As such, the decentralized method can scale to much larger groups of robots, particularly when each robot has complicated dynamics. However, these decoupled methods fundamentally cannot be complete, and so may fail to find a solution even if one exists. Other work that uses sensing

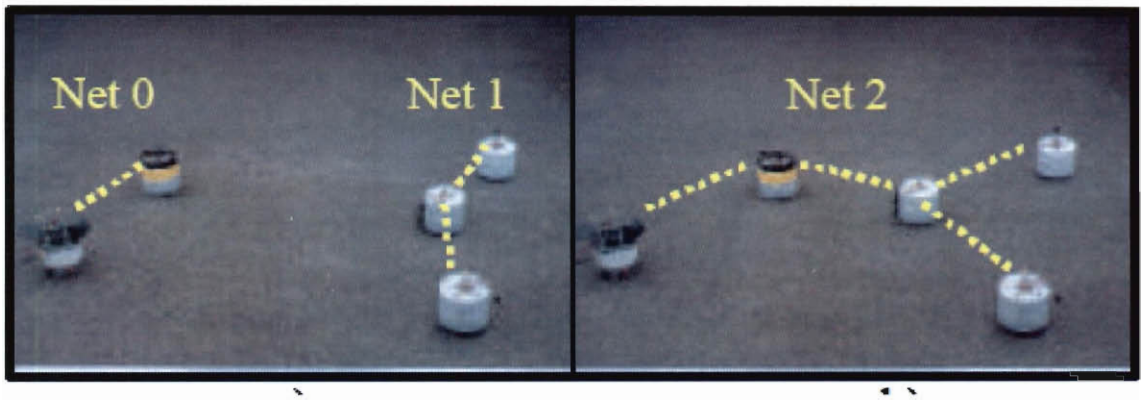


Figure 2.2: An example of the hybrid centralized, distributed approach [57] based on dynamic networks.

and communication to coordinate a team of cooperative robots [61] assumes that collision avoidance is easy using a potential function that pushes away from obstacles and neighboring robots. However, it does not address the ICS problems that may occur with complex dynamics, and only communicates to optimize the goal of target tracking, rather than for safety.

Additionally, assuming that all communication is strictly between robots in a limited range, with centralized schemes eventually multi-hop communication would be needed. A completely decentralized focus alleviates the need to consider the many challenges that ad-hoc multi-hop networking poses [62, 63, 64].

2.4 Safe, Coordinated Navigation

This thesis extends earlier work [2], where integration of a kinodynamic sampling-based planner with ICS avoidance schemes were applied [25] to safely plan for multiple

robots that formed a network and explored an unknown environment. The existing work follows a decoupled approach, but in contrast to velocity tuning, it weakly constrains the motion of a robot before considering their interactions since it considers multiple alternative paths for each robot at each cycle. At the same time, it does not impose a predefined priority on the robots but instead robots respect their neighbors in a way that emerges naturally from their unsynchronized operation. In addition, the robots in previous work followed a synchronous planning operation, based on a global clock, which simplified the coordination process, although this work continues assuming clocks that progress at the same rate. Both previous work and this work assume that there is a well-defined radius of perfect communication.

Finally, it is worth noting that we are developing a cooperative system. It is assumed that all robots want to be safe and will follow at least the basic protocol. Adversarial robots are not addressed here. There are many approaches [65, 66, 67] from competitive robotics literature to draw from. Although the problem they consider is largely distinct, some of the reasoning may be applicable. With communication failures between robots, a robust consensus algorithm is required, for which a survey paper [68] is useful for finding an algorithm that would fit our needs. Although these issues would need solutions in a real-world large scale deployment, this is not the focus of this thesis. Instead, a general protocol is developed and it is hoped that future research would consider these challenges prior to any such deployment.

Chapter 3

A Safe Solution to Distributed Motion Planning

3.1 Problem Statement

Consider robots operating in the same known workspace with static obstacles. Each **robot** R^i exhibits drift over all but a measure 0 subset of the state space and must satisfy non-holonomic constraints. Drift is the characteristic that the system state will not remain constant over time in the case of zero control input. The constraints are expressed by differential equations of the form: $\dot{x}^i = f^i(x^i, u^i)$, $g^i(x^i, \dot{x}^i) \leq 0$, where $x^i \in \mathcal{X}^i$ represents a **state**, u^i is a **control** and f^i, g^i are smooth. The subset of the **state space** $\mathcal{X}^i$ that does not cause a collision with static obstacles is denoted as $\mathcal{X}^i_f$. The robot model used in this thesis can be found in Section 4 and involves acceleration controlled car-like systems, including versions with minimum positive velocity. The robot shape is defined as the union of arbitrary poly-

gons and is used for collision checking against the environment. Collision checking against other robots is done only with a safety radius for computational efficiency. All robots use a global coordinate frame, because without sensing of some shared feature, such as relative positions of the robots, or transmission of at least one global coordinate transformation, it would be impossible to align local coordinate frames. This issue is not considered because addressing it would either involve addition of a specific sensor model, which is not desired, or simply applying a coordinate transform, which is trivial. All robots share an identical static map of the environment, so given that we are assuming good localization, it seems reasonable to assume a global coordinate reference.

Each R^i is located at an initial state $x^i(0)$ and must compute plans that will bring it to its individual goal $x_g^i(t_{max})$ without collisions and within finite time t_{max}. This finite time assumption cannot be guaranteed because of the lack of completeness and liveness guarantees, however, it has never been experimentally violated, given a large enough t_{max}. Then:

- A **plan** is a sequence of controls $p(dt) = \{(u_1, dt_1), \ldots, (u_n, dt_n)\}$ $(dt = \sum_i dt_i)$.
- A plan $p(dt)$ executed at state $x(t)$ defines a **trajectory**: $\pi(x(t), p(dt))$, which is a sequence of states.
- A trajectory is **feasible** as long as it satisfies functions f^i and g^i for robot R^i.
- A plan $p(dt)$ is **valid** at state $x(t)$, if it defines a feasible trajectory $\pi(x(t), p(dt))$.
- A state along $\pi(x(t), p(dt))$ at time $t' \in [t : t + dt]$ is denoted as $x[\pi(x(t), p(dt))](t')$.
- A feasible trajectory $\pi(x(t), p(dt))$ is **collision-free** with respect to the static obstacles if: $\forall\, t' \in [t : t + dt] : \quad x[\pi(x(t), p(dt))](t') \in \mathcal{X}_f$.
- For a **trajectory concatenation** $\pi'(\pi(x(t), p(dt)), p'(dt'))$, plan $p(dt)$ is executed at $x(t)$

and then $p'(dt')$ is executed at state: $x[\pi(x(t), p(dt))](t + dt)$.

- Two trajectories for robots R^i and R^j are **compatible:**

$$\pi^i(x^i(t^i), p(dt^i)) \asymp \pi^j(x^j(t^j), p^j(dt^j)) \text{ as long as}$$

$$x[\pi^i](t) \asymp x[\pi^j](t) \quad \forall\ t \in [max(t^i, t^j) : min(t^i + dt^i, t^j + dt^j)]$$

where $x^i \asymp x^j$ means that R^i in state x^i does not collide with R^j at state x^j. The corresponding plans $p(dt^i)$, $p(dt^j)$ are also called compatible at states $x^i(t^i)$, $x^j(t^j)$.

The robots are equipped with an omnidirectional, range-limited communication ability, which is assumed to be reliable and can be utilized for coordination and pairwise collision avoidance. The set of all robots within communication range of R^i is called the neighborhood N^i. A robot does not have any information about any other robot unless they communicate.

Given the above notation, the problem of **distributed motion planning with dynamics** (DMPD) can be defined as follows: Consider m robots with range-limited communication capabilities operating in the same workspace with obstacles. Each robot's motion is governed by second-order dynamics specified by f^i and g^i. Initially, robot R^i is located at state $x^i(0)$, where $x^i(0) \in \mathcal{X}^i_f$ and $\forall i, j :\ x^i(0) \asymp x^j(0)$. Each R^i must compute a valid plan $p^i(t_{max})$ so that:

- $x[\pi^i(x^i(0), p^i(t_{max}))](t_{max}) = x^i_g(t_{max})$ (i.e., the plans bring the robots to their individual goals within time t_{max}),
- $\forall\ i,\ \ \forall t \in [0 : t_{max}] :\ \ x[\pi^i(x^i(0), p^i(t_{max}))](t) \in \mathcal{X}_f$ (i.e., the resulting trajectories are collision-free with static obstacles)

- and $\forall\ i, j\ :\quad \pi^i(x^i(0), p^i(t_{max})) \asymp \pi^j(x^j(0), p^j(t_{max}))$ (i.e., the trajectories are pairwise compatible from the beginning and until all the robots reach their goals).

3.2 A Simple Framework without Safety Guarantees

As discussed in Chapters 1 and 2, this thesis adopts a decentralized framework for scalability purposes. Instead of velocity tuning and fixed prioritization, the robots coordinate on the fly within a replanning framework. Each robot's operation is broken into intervals $([t_0^i : t_1^i], [t_1^i, t_2^i], \ldots, [t_n^i : t_{n+1}^i], \ldots)$, called cycles. During cycle $[t_{n-1}^i : t_n^i]$, robot R^i considers multiple alternative plans Π^i for the next cycle $[t_n^i : t_{n+1}^i]$, given the future initial state $x^i(t_n^i)$. Through coordination, R^i selects plan $p_*^i([t_n^i : t_{n+1}^i])$.

It is assumed that the duration of each cycle is constant and the same for all robots: $\forall i,\ \forall n\ :\ t_{n+1}^i - t_n^i = dt$. Nevertheless, the robots are unsynchronized: the cycles among different robots do not coincide and t_0^i is typically different than t_0^j. Synchronicity is a restrictive assumption, as it requires all the robots to initiate their operation at exactly the same time although they may be located in different parts of the world and may not communicate their initial states. In fact, given a limited communication range, it may not be the case that the robots form a connected communication graph, and so it would not be possible to synchronize all robots.

Given this setup, Algorithm 3.2.1 outlines a straightforward approach for the single cycle operation of each robot that tries to find compatible plans. During $[t_{n-1}^i : t_n^i]$, R^i computes alternative partial plans Π^i for the consecutive planning cycle. In parallel, R^i listens for messages from robots in neighborhood N^i that contain their selected trajectories.

Algorithm 3.2.1 Simple but Unsafe Operation of R^i During Cycle $[t^i_{n-1} : t^i_n]$

$\Pi^i \leftarrow \emptyset$ and $\Pi^{N^i} \leftarrow \emptyset$
while $t < t^i_n - \epsilon$ **do**
  $\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})) \leftarrow$ collision-free trajectory from a single-robot planner
  $\Pi^i \leftarrow \Pi^i \cup \pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1}))$
  if $R^j \in N^i$ is transmitting a trajectory π^j **then**
    $\Pi^{N^i} \leftarrow \Pi^{N^i} \cup \pi^j$
  end if
end while
for all $\pi^i \in \Pi^i$ **do**
  for all $\pi^j \in \Pi^{N^i}$ **do**
    if $\pi^i \not\asymp \pi^j$ (incompatible trajectories) **then**
      $\Pi^i \leftarrow \Pi^i - \pi^i$
      break
    end if
  end for
end for
$\pi^i_* \leftarrow$ trajectory in Π^i which brings R^i closest to the goal
Transmit π^i_* to all neighbors in N^i and execute π^i_* during next cycle

When time approaches $t^i_n - \epsilon$, R^i selects among all trajectories that are collision-free and compatible with the neighbors' messages, the one that brings the robot closer to its goal. If such a trajectory is indeed found at each iteration, then the DMPD problem is eventually solved by this algorithm.

3.3 Ensuring Safety

A robot following the above approach might fail to find a trajectory π^i_* because the set Π^i might be empty, which means that every considered trajectory will lead to a collision. This section describes a distributed algorithm that guarantees the existence of a collision-free,

compatible trajectory for all robots at every cycle.

3.3.1 Safety Considerations: Inevitable Collision States

Although the robot R^i may not be in ICS, it may still be the case that the planning process did not find any feasible paths. If robot R^i is in ICS, then no matter what planning process we performed, we are guaranteed to find that Π^i is empty. State $x(t)$ is ICS with regards to static obstacles if:

$$\forall\ p(\infty):\ \exists\ dt \in [t, \infty) \text{ so that } x[\pi(x(t), p(\infty))] \notin \mathcal{X}_f.$$

Computing whether a state is ICS is intractable, since it requires reasoning over an infinite horizon for all possible plans. It is sufficient, however, to consider conservative methods that identify states that are *not* ICS [22, 25]. The approximation reasons over a subset of predefined maneuvers $\Gamma(\infty)$, called here **contingency plans**. If R^i can avoid collisions in the future with static obstacles at $x^i(t_n)$ by guaranteeing that a contingency plan $\gamma^i(\infty) \in \Gamma^i(\infty)$ avoid collisions over an infinite horizon, then $x^i(t_n)$ is not ICS with regards to static obstacles. For cars, braking maneuvers are sufficient since it is possible to reason over an infinite time horizon whether these plans will collide with static obstacles. Circling maneuvers can be used for systems with minimum velocity limits, such as airplanes. So, robots can avoid ICS by explicitly checking to make sure that there is always at least one $\infty - time$ feasible path at the end of our partial plans.

Multiple moving robots pose new challenges for ICS. Trajectories π^i and π^j may be compatible for the next cycle, but the corresponding robots may reach states that will in-

evitably lead them in a future collision. Thus, safety notions have to be extended to the multi-robot case. It is still necessary for computational reasons to be conservative and focus only on a set of contingency plans. For m robots $\{R^1, R^2, \ldots, R^m\}$ executing plans $\{p^1(dt^1), p^2(dt^2), \ldots, p^m(dt^m)\}$ at states $\{x^1(t), x^2(t), \ldots, x^m(t)\}$, state $x^i(t)$ is considered a **safe state** if:

$$\exists\, \gamma^i(\infty) \in \Gamma^i(\infty) \text{ so that } \forall\, t' \in [t, \infty) : \quad x[\pi^i(x^i(t), \gamma^i(\infty))](t') \in \mathcal{X}_f$$

and

$$\forall\, j \in [1, m],\ j \neq i,\ \exists\, \gamma^j(\infty) \in \Gamma^j(\infty) : \quad \pi^i(x^i(t), \gamma^i(\infty)) \asymp \pi^j(\, \pi^j(x^j(t), p^j(dt^j)),\ \gamma^j(\infty)).$$

Or to be more clear,

from the state in question, there is some contingency plan $\gamma^i(\infty)$ from the set of possible contingency plans $\Gamma^i(\infty)$, so that from the time that state is entered, to unbounded time in the future, the states that comprise the trajectory of this contingency plan are in the free part of the workspace,

and

for all other robots j, their stated intentions, appended with their contingency plans, are compatible trajectories with respect to our contingency plan.

Therefore our contingency check should be to, from a set of contingency plans, first check

against static obstacles, then, of those that pass, check against the plans of other robots, and always check against their possible contingencies as well, to ensure that all robots have at least one safe option at this state.

In the above definition, dt^j is the duration of robot R^j's cycle. Note that a trajectory concatenation is used for R^j's trajectory. In this trajectory concatenation, $p^j(dt^j)$ is executed for time dt^j and then the contingency $\gamma^j(\infty)$ is applied. The reason is that, as robots decide in an unsynchronized fashion, it may happen that at t, robot R^j has already committed to plan $p^j(dt^j)$. Extending the assumption in the problem statement about compatible starting states, the following discussion will assume that the initial states of all the robots are safe states. Then an algorithm for the DMPD problem must maintain the following invariant for each robot and planning cycle:

Safety Invariant: The selected trajectory $\pi^i_*(x^i(t^i_n), p^i(t^i_n : t^i_{n+1}))$:

a) Must be collision-free with obstacles.

b) Must be compatible with all other robots, during the cycle $(t^i_n : t^i_{n+1})$:

$$\pi^i_*(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})) \asymp \pi^j_*(x^j(t^i_n), p^j(t^i_n : t^i_{n+1})), \quad \forall j \neq i$$

c) And the resulting state $x[\pi^i_*](t^i_{n+1})$ is safe for all possible future plans $p^j(t^i_{n+1} : \infty)$ selected by other robots ($j \neq i$). In other words, the concatenation of trajectory π^i_* with a contingency $\gamma^i(\infty)$ must be compatible with the concatenations of trajectories $\pi^j_*(dt)$ of other vehicles with their contingencies $\gamma^j(\infty)$:

$$\forall j \neq i$$

$$\pi^i(\pi^i_*(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})), \gamma^i(\infty)) \asymp$$
$$\pi^j(\pi^j_*(x^j(t^i_n), p^j(t^i_n : t^i_{n+1})), \gamma^j(\infty))$$

Point c) above means that R^i has a contingency plan at $x[\pi^i_*](t^i_{n+1})$, which can be safely followed for the other robots' choices given the algorithm. If the invariant holds for all the robots, then they will always be safe. If for any reason a robot cannot find a plan that satisfies these requirements, then it can revert to its contingency that guarantees its safety.

3.4 A Safe, Unsynchronized and Decentralized Solution

Algorithm 3.3.1, in contrast to Algorithm 3.2.1, maintains the safety invariant. The protocol follows the same high-level framework and still allows a variety of planning techniques to be used for producing trajectories. The differences with the original algorithm can be summarized as follows:

- The algorithm stores the messages received from neighbors during the previous cycle in the set $\Pi^{N^i}_{prev}$ *(lines 1-3)*. Note that the robots transmit the selected trajectory together with the corresponding contingency *(lines 12-13 and 22)*.
- A contingency plan $\gamma(t^i_{n+1} : \infty)$ is attached to every collision-free trajectory $\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1}))$ and the trajectory concatenation π^i_γ is generated *(line 5-6)*. Note that potentially multiple different contingencies can be attached to the trajectory $\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1}))$. Each resulting trajectory concatenation is treated individually by the algorithm.
- The trajectory π^i_γ is added to Π^i only if it is collision-free with static obstacles for an infinite time horizon *(lines 7-8)*, thus guaranteeing that $x[\pi^i](t^i_{n+1})$ is not ICS.
- π^i_γ is rejected, however, if it is not compatible with all the trajectories and contingencies

Algorithm 3.3.1 Safe and Unynchronized Operation of R^i During Cycle $[t^i_{n-1} : t^i_n]$

$\Pi^i \leftarrow \emptyset$, $\Pi^{N^i}_{prev} \leftarrow \emptyset$, $\Pi^{N^i}_{new} \leftarrow \emptyset$
for all $R^j \in N^i$ **do**
  $\Pi^{N^i}_{prev} \leftarrow \Pi^{N^i}_{prev} \cup \pi^j(\, \pi^j(x^j(t^j_{n-1}), p^j(t^j_{n-1} : t^j_n)),\ \gamma(t^j_n : \infty)\,)$
  (i.e., include all past trajectories and attached contingencies of neighbors)
end for
while $t < t^i_n - \epsilon$ **do**
  $\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})) \leftarrow$ collision-free trajectory from a single-robot planner
  $\pi^i_\gamma \leftarrow \pi^i(\, \pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})),\ \gamma(t^i_{n+1} : \infty))$ (i.e., contingency concatenation)
  if $\forall\, t \in [t^i_{n+1} : \infty) :\ x[\pi^i_\gamma](t) \in \mathcal{X}_f$ **then**
    $\Pi^i \leftarrow \Pi^i \cup \pi^i_\gamma$
    for all $\pi^j_\gamma \in \Pi^{N^i}_{prev}$ **do**
      if $\pi^i_\gamma \not\asymp \pi^j_\gamma$ **then**
        $\Pi^i \leftarrow \Pi^i - \pi^i_\gamma$
      end if
    end for
  end if
  if $R^j \in N^i$ is transmitting a trajectory and an attached contingency **then**
    $\Pi^{N^i}_{new} \leftarrow \Pi^{N^i}_{new} \cup \pi^j_\gamma(\, \pi^j(x^j(t^j_n), p^j(t^j_n : t^j_{n+1})),\ \gamma(t^j_{n+1} : \infty)\,)$
  end if
end while
for all $\pi^i_\gamma \in \Pi^i$ **do**
  for all $\pi^j_\gamma \in \Pi^{N^i}_{new}$ **do**
    if $\pi^i_\gamma \not\asymp \pi^j_\gamma$ **then**
      $\Pi^i \leftarrow \Pi^i - \pi^i_\gamma$
    end if
  end for
end for
if Π^i empty or if a message was received during compatibility check **then**
  $\pi^i_* \leftarrow \pi^i(x^i(t^i_n), \gamma(t^i_n : \infty))$ (i.e., follow the available contingency for next cycle)
else
  $\pi^i_* \leftarrow$ trajectory in Π^i which brings R^i closer to the goal given a metric
end if
Transmit π^i_* to all neighbors in N^i and execute π^i_* during next cycle

of neighbors from the set $\Pi^{N^i}_{prev}$.

- During the compatibility check *(lines 14-17)*, R^i checks not just trajectories for the next cycle but its trajectory concatenations with contingencies π^i_γ against its neighbors' trajectory concatenations π^j_γ.
- The final change *(lines 18-21)* addresses the case that Π^i is empty or when a message arrives while R^i executes its compatibility check. If any of the two is true, then R^i selects to follow the contingency $\gamma(t^i_n : \infty)$, which was used in the previous cycle to prove that $x(t^i_n)$ was safe. Otherwise, R^i selects among the set Π^i the trajectory that brings it closer to the goal according to a desired metric. previous cycle, stored in $\Pi^{N^i}_{prev}$ *(lines 9-11)*.
- The while loop *(lines 4-13)* is executed as long as time t is less than the end of the planning cycle (t^i_n) minus an ϵ time period. Time ϵ should be sufficient for the robot to complete the compatibility check *(lines 14-17)* and the selection process *(lines 18-22)*. If the robot is running out of time, the robot should immediately select a contingency in order to guarantee safety. In a real robot implementation, this can be achieved through an interrupt or a signal that stops execution and enforces the contingency. In a serial implementation ϵ has to be sufficiently large, or it will always run out of time and execute a contingency plan. ϵ is dependent on several factors, primarily resolution of collision checking, computational power, and message latency (latency issues are introduced in Section 3.7).

Overall, each robot selects a plan $p^i(t^i_n : t^i_{n+1})$ and contingency $\gamma^i(t^i_{n+1} : \infty)$ that respect the plans and contingencies of other robots that have been selected before time t^i_n. If no such plan is found or there is no time to check against newly incoming messages, then the contingency $\gamma^i(t^i_n : \infty)$ is selected.

3.5 Computational Complexity

The complexity of the algorithm depends on the number of neighboring robots N^i, which in the worst case it corresponds to the total number of robots, minus the self, or $N - 1$. To make the equations cleaner, simply N is used, as the difference is negligible in large teams. In order to represent the cost of operations involving trajectories, it is important to consider a representation for a trajectory. One way to represent a trajectory is through a discrete sequence of states, which are selected given a predefined resolution in time Q (i.e., the technique becomes resolution-safe in this case). Resolution as defined by space is also possible, but then the message length would change depending on state parameters such as velocity. Using time resolution, we have a constant-size message for a given plan (of uniform cycle length time). Then, let us denote as S the upper limit in the number of states used to represent each trajectory . For the case of braking maneuvers, this upper limit can be calculated by:

$$S = \frac{dt}{Q} + \frac{\frac{v_{max}}{\alpha}}{Q}$$

The first term in the summation corresponds to the number of states used to represent a collision-free trajectory for a planning cycle of duration dt. The second term is the upper limit in the number of states needed to represent a braking maneuver, where v_{max} is the maximum velocity of the system and α its maximum deceleration. Let's denote with P, the upper limit in the number of plans considered during each planning cycle for the current agent.

Given the above notation, the complexity of the algorithm's various operations is as follows:

(a) *Lines 2-4:* $S \times N$,

(b) *Lines 6 - 9:* $P \times S$,

(c) *Lines 10 -14:* $P \times N \times S^2$ (if the states in a trajectory are not accompanied by a global timestamp) or $P \times N \times S$ (if the states are tagged with a global timestamp),

(d) *Lines 16-18:* $S \times N$,

(e) *Lines 20-26:* $P \times N \times S^2$, (if the states in a trajectory are not accompanied by a global timestamp) or $P \times N \times S$ (if the states are tagged with a global timestamp),

(f) *Lines 27-31:* P, assuming constant time for computing a cost-to-go metric for each state,

(g) *Line 32:* $N \times S$.

Overall, the worst-case complexity for an unsynchronized system that lacks global timestamps is: $P \times N \times S^2$. Note that for robots with limited communication, the parameter N is reduced. Typically Q (which determines S) should be small (high temporal resolution) to not introduce collisions due to resolution issues. Lower maximum velocity or higher maximum deceleration also assist computationally, in the specific case of braking maneuvers. Similarly, considering fewer plans reduces computational complexity but reduces the diversity of solutions considered at each time step.

3.6 Guaranteeing Maintenance of the Safety Invariant

This section provides a proof that Algorithm 3.3.1 maintains the safety invariant of Section 3.3.1 given some simplifying assumptions that will be waived later in Section 3.7.

Theorem 1: Algorithm 3.3.1 guarantees the maintenance of the safety invariant of Section 3.3.1 (which shall now be referred to as simply the Invariant) in every planning cycle given it holds during the cycle $(t_0^i : t_1^i)$ and the simplifying assumptions that:

i) all robots can communicate one with another

ii) plans are transmitted instantaneously between robots.

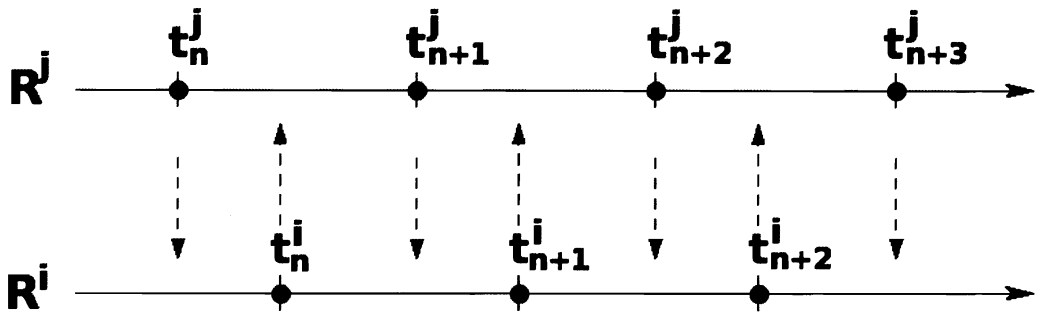


Figure 3.1: The replanning cycles of two neighboring robots R^i and R^j. The times denote transitions between planning cycles for each robot. The vertical arrows denote the transmission of information, e.g., at t_n^i, R^i transmits $\pi^i(\ \pi^i(x^i(t_n^i), p^i(t_n^i : t_{n+1}^i)),\ \gamma(t_{n+1}^i : \infty)\ )$.

Proof: The proof is obtained by induction. The *base case* holds for R^i because of the Theorem's assumption that the Invariant holds during cycle $(t_0^i : t_1^i)$. The *inductive step* will show that if the Invariant holds during the cycle $(t_n^i : t_{n+1}^i)$ then it will also hold during the cycle $(t_{n+1}^i : t_{n+2}^i)$ for Algorithm 3.3.1. Without loss of generality consider Figure 3.1 and focus on robot R^i. To prove the inductive step, it is necessary to show that each one of

the three points of the Invariant will be satisfied during $(t^i_{n+1} : t^i_{n+2})$. For cycle $(t^i_{n+1} : t^i_{n+2})$ there are two cases: (1) A compatible trajectory $\pi^i_* = \pi^i_\gamma \in \Pi^i$ is selected, or (2) the current contingency is returned.

Case 1: A trajectory $\pi^i_\gamma \in \Pi^i$ is selected.

a) Trajectory π^i_γ has to be collision-free as part of Π^i.

b) Assuming instantaneous plan transmission and by time t^i_{n+1}, R^i has available the choices of other robots for cycles that start before t^i_{n+1}. Since $\pi^i_\gamma \in \Pi^i$ is selected, none of this message arrived during the compatibility check. This means that R^j's trajectory $\pi^j(\ \pi^j(x^j(t^j_{n+1}), p^j(t^j_{n+1} : t^j_{n+2})),\ \gamma(t^j_{n+2} : \infty)\ )$ is available to R^i during the compatibility check. Then the cycle $(t^i_{n+1} : t^i_{n+2})$ can be broken into two parts:

i) During part $(t^i_{n+1} : t^j_{n+2})$, the selected plan $p^i(t^i_{n+1} : t^i_{n+2})$ is compatible with $p^j(t^j_{n+1} : t^j_{n+2})$ because the second plan was known to R^i when selecting π^i_γ.

ii) For part $(t^j_{n+2} : t^i_{n+2})$ there are two cases for R^j at time t^j_{n+2}:

- R^j will either select a plan $p^j(t^j_{n+2} : t^j_{n+3})$ that is compatible with $p^i(t^i_{n+1} : t^i_{n+2})$,
- or it will resort to a contingency $\gamma^j(t^j_{n+2} : \infty)$, which, however, is already compatible with trajectory π^i_γ.

In both cases, R^j will follow a plan that is compatible with $p^i(t^i_{n+1} : t^i_{n+2})$.

Thus, the second point b) of the Invariant is also satisfied for robots R^i and R^j.

c) For the third point of the Invariant, the contingency $\gamma^i(t^i_{n+2} : \infty)$ has to be compatible with the future choices of the other robots. Focus again on the interaction between R^i and R^j. There are again two cases for R^j at time t^j_{n+2}:

i) R^j will select a plan $p^j(t^j_{n+2} : t^j_{n+3})$ and a corresponding contingency $\gamma^j(t^j_{n+3} : \infty)$. This

plan and contingency respect by construction R^i's contingency $\gamma^i(t^i_{n+2} : \infty)$, since it was known to R^j at time t^j_{n+2}.

ii) Or R^j will resort to its contingency $\gamma^j(t^j_{n+2} : \infty)$, which, however, the contingency $\gamma^i(t^i_{n+2} : \infty)$ respected upon its selection.

In any case, whatever R^j chooses at time t^j_{n+2}, it is going to follow plans in the future that are compatible with $\gamma^i(t^i_{n+2} : \infty)$. Thus, point c) is also satisfied.

Case 2: A contingency $\gamma^i(t^i_{n+1} : \infty)$ was selected.

The *inductive hypothesis* implies that $x^i(t^i_{n+1})$ is a *s*afe state. Thus:

a) $\gamma^i(t^i_{n+1} : t^i_{n+2})$ is collision-free with static obstacles

b) The current plans of all robots will be compatible with $\gamma^i(t^i_{n+1} : t^i_{n+2})$, which was known to them at time t^i_n. Furthermore, $\gamma^i(t^i_{n+1} : t^i_{n+2})$ already respects the contingencies of other robots that might be executed before t^i_{n+1}.

c) The state $x^i[\gamma^i(t^i_{n+1} : \infty)](t^i_{n+2})$ is trivially safe, because R^i can keep executing the same contingency for ever and this contingency will have to be respected by its neighbors, as it will always be known ahead of time.

So, in both cases, all three points of the Invariant are satisfied for R^i and the inductive step is proved. Thus, if the Invariant holds, the algorithm maintains its validity. ■

3.7 Addressing the Assumptions

Theorem 1 assumed that messages are transmitted instantaneously and that all the robots communicate one with another. The assumption that plans are transmitted instantaneously will not hold in real-world experiments with wireless communication. Similarly,

it is more realistic to assume that robots can communicate only if their distance is below a certain threshold. In the latter case, the proposed approach can be invoked using only point to neighborhood communication and thus achieve higher scalability. The following theorem shows that the safety guarantees can be provided without these restrictive assumptions.

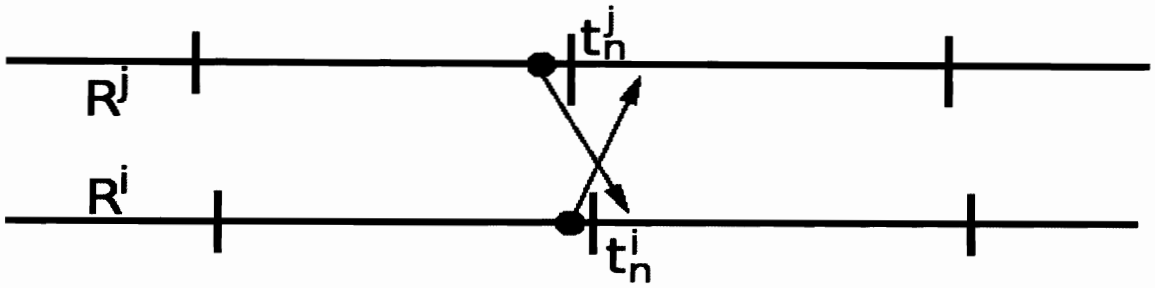


Figure 3.2: If messages arrive after the start of a neighbor's future cycle, as with the message from R^j to R^i above, this is problematic, but can be addressed.

Theorem 2: Algorithm 3.3.1 guarantees the maintenance of the Invariant in every planning cycle given it holds during cycle $(t_0^i : t_1^i)$ and that:

i) two robots with limited communication ranges can communicate before they enter into ICS given a predefined set of contingencies $\Gamma(\infty)$.

ii) robots utilize acknowledgments that signal the reception of a trajectory by a neighbor.

Sketch of Proof: Theorem 1 showed that the invariant holds as long as it was valid during the first cycle $(t_0^i : t_1^i)$ and that two vehicles are able to communicate continuously since time t_0^i. For two robots with limited communication range, denote as time t_{comm} the beginning of the first planning cycle of either robot after they are able to communicate. If at t_{comm}, both robots have available a contingency $\gamma(\infty) \in \Gamma(\infty)$, that can be used to prove the

safety of their corresponding states, then all the requirements of Theorem 1 are satisfied for $t_0^i = t_{comm}$. Thus the invariant will be maintained. The discussion after this proof, shows an example of how it is possible to guarantee the existence of such a contingency for the case of car-like vehicles using braking maneuvers for contingencies.

Regarding the issue of delayed transmission of trajectories, consider the case that R^j's cycle ends at time t_n^j, which is before the end of the neighboring R^i's cycle at time t_n^i. Figure 3.2 provides an example. If the transmission of the trajectory π_*^j to R^i is delayed, it might arrive after time t_n^i and R^i cannot detect that it did not take into account the choice of R^j during its compatibility check given Algorithm 3.3.1. Thus, R^i's choice might end up being incompatible with π_*^j. Notice that this problem becomes more frequent when Algorithm 3.3.1 is employed by robots that have synchronized cycles. If an acknowledgment message that signals the reception of a trajectory by a neighbor is used, however, R^i can acknowledge the message's reception, whether it arrives before or after t_n^i. If the acknowledgment arrives at R^j before t_n^j (as well as from all other neighbors), it knows that it is safe to execute π_*^j. If the acknowledgment is not received on time, R^j can revert to its contingency which is by construction respected by the future plan of R^i, whatever this is. Thus, the introduction of an acknowledgment resolves the issue of possible delays in the transmission of trajectories.

■

As an example of robots with limited communication range, consider a worst case situation between two car-like vehicles, which are located right outside their communication ranges and head one towards the other with maximum velocity. For braking maneuvers as contingencies, it is possible to limit the maximum velocity that the systems can achieve so as to guarantee that they always have enough time to decelerate at the point of first commu-

nication. In particular, the maximum velocity of the car-like system v_{max} has to be limited as follows:

$$v_{max} < \left(\sqrt{\frac{4\mathbb{C}^2 + (\mathbb{R} - \mathbb{S})}{\alpha}} - 2\mathbb{C} \right) \alpha$$

where $\mathbb{C}$ is the duration of a planning cycle, $\mathbb{R}$ is the communication range of the robots, $\mathbb{S}$ is the greatest dimension of the size of the robots, and α is the maximum acceleration/deceleration the robots can sustain. This equation can be transformed to provide the minimum communication range of the robot required for a given maximum velocity, depending on the design of the system in question. Similar equations can be produced for different systems and different contingency plans, such as periodic circling maneuvers. For these systems, the circling distance in any applicable direction is needed, as well as a calculation of the maximum time to circle back to the start of the periodic motion.

For the full algorithms, see Appendix A.

Chapter 4

Implementation

In order to validate the theoretical discussion, simulations were conducted. These experiments explore the boundary cases of the algorithm and suggest the type of physical platform to use for extended validation. Additionally, initial experiments revealed performance deficits and suggested strategies for improvement. Based on these initial experiments, practical modifications in the implementation of the algorithm led to significant speed ups and quick convergence to a solution.

Modeled System: The experiments presented in this thesis are using the model of a second-order car like vehicle [69] shown on the right side, where (x, y) are the car's reference point Cartesian coordinates, θ is the car's orientation, w its velocity and ζ the steering angle. The

controls are α, the acceleration, and ϕ the rate of change of the steering angle.

$$\dot{x} = w \cdot cos(\zeta) \cdot cos(\theta)$$

$$\dot{y} = w \cdot cos(\zeta) \cdot sin(\theta)$$

$$\dot{\theta} = w \cdot sin(\zeta)$$

$$\dot{w} = \alpha$$

$$\dot{\zeta} = \phi$$

There are limits both for state and control parameters:

$$|w| < w_{max}$$

$$|\zeta| < \zeta_{max}$$

$$|\alpha| < \alpha_{max}$$

$$|\phi| < \phi_{max}$$

All robots have range-limited communication, out to 30% of the total environment width, and brake to zero speed for contingency. Given a planning cycle time, $\mathbb{C}$, communication range, $\mathbb{R}$, and the control inputs, we compute w_{max} as

$$w_{max} = \left(-2 \cdot \mathbb{C} + \sqrt{4 \cdot \mathbb{C}^2 + \mathbb{R}/\alpha}\right) \cdot \alpha$$

Environments Four simulated environments were used for the experiments:

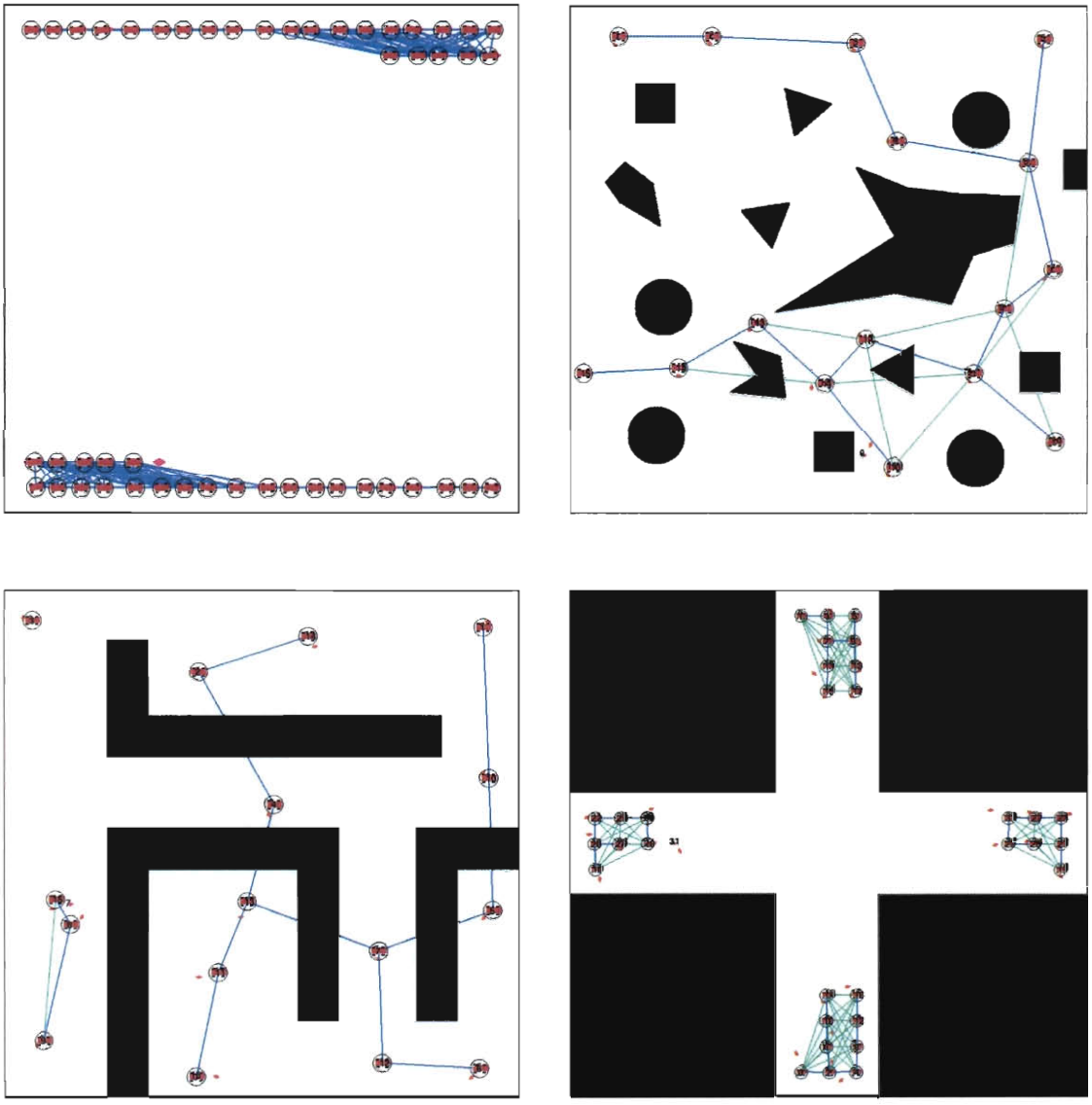


Figure 4.1: Starting positions for the environments simulated.

1. An "(E)mpty" environment (Fig. 4.1, top left, shown with 48 robots),
2. a "(R)andom" environment containing 14 polygonal obstacles of various shapes and sizes (Fig. 4.1, top right, shown with 16 robots),

3. an "(O)ffice" environment consisting mainly of fairly wide corridors and open rooms (Fig. 4.1, bottom left, shown with 16 robots), and

4. an "(I)ntersection" environment with two crossing corridors (Fig. 4.1, bottom right, shown with 32 robots).

These environments are presented in approximate order of difficulty. The various experiments tested different numbers of vehicles: 2, 4, 8, 16, 32, 48. The size of the robots was reduced to half for the 32 robot case, and to a quarter of their size for the 48 robot case. If this was not done, then the robots would take up 12% and 18% of the workspace, respectively. Since much of the workspace is already occupied by obstacles, this reduction in size assists in reducing clutter effects that result in higher solution time.

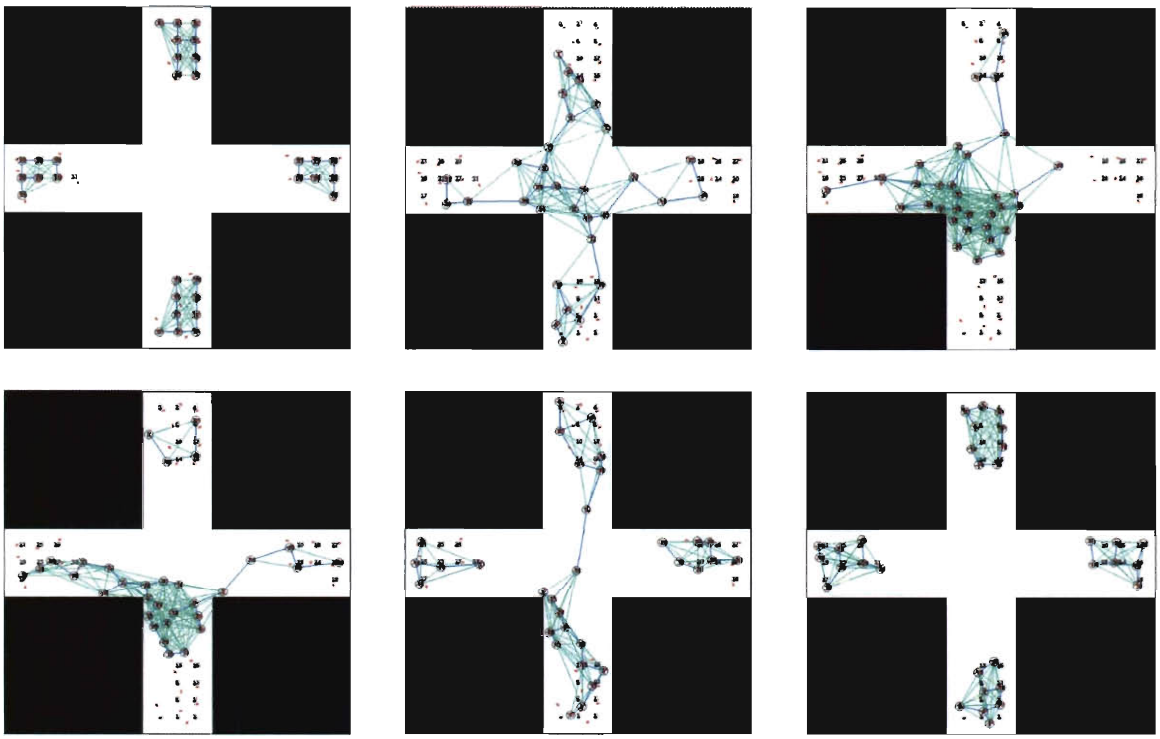


Figure 4.2: Snapshots from a typical run in the intersection environment with 32 robots.

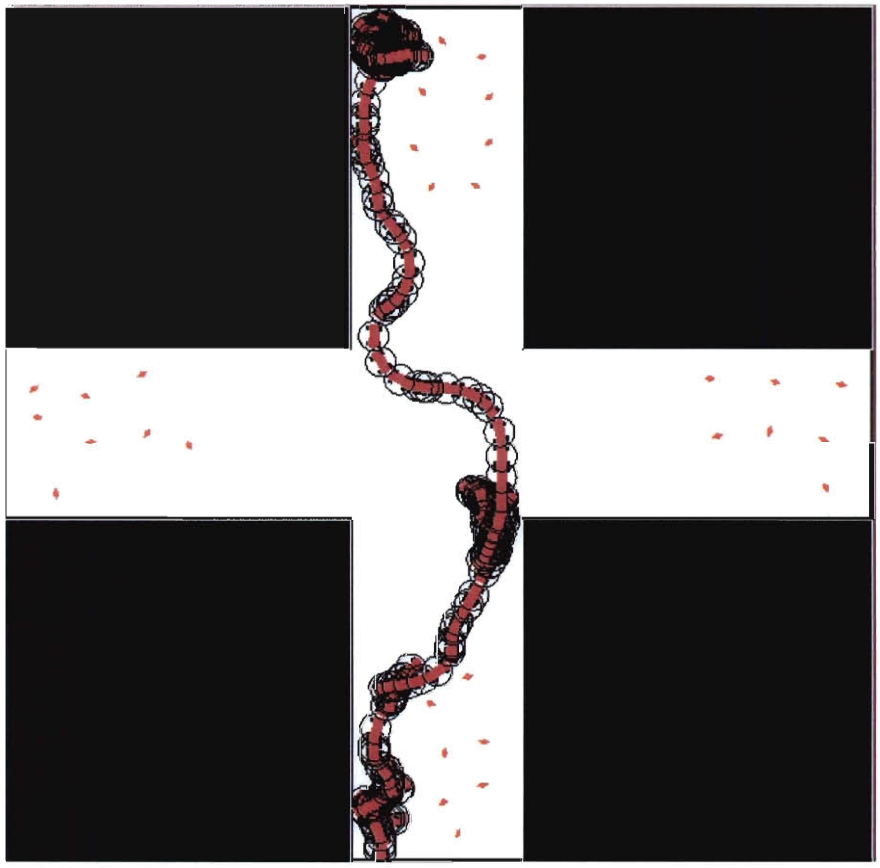


Figure 4.3: The full trajectory of robot 0 - diamonds represent actual final states of all robots.

The empty environment was the easiest to solve. The office environment was chosen as a gauge for how hard a structured environment can be. The robots, in their original size, are about 1/5 of the size of the hallway. In the random environment, there were polygons of varying shapes and sizes. The intersection case seemed to be the hardest to solve, since the robots not only have to navigate through a relatively narrow passage together with their neighbors, but they are all forced to traverse the center, almost simultaneously.

Where possible, starting/goal locations were kept the same across runs as more robots

were added. Experiments for the same number of robots have the same start/goal locations. All experiments were repeated at least 10 times. In each test, the algorithm ran in real time such that computation time is equal to execution time.

4.1 Implementation Specifics

The algorithm, exactly as implemented for the results in Chapter 4, is Algorithm A.1.1. This section highlights some steps taken to make the implementation of Algorithm 3.3.1 more efficient computationally. In particular:

- Instead of checking the compatibility of all the candidate plans Π^i with the transmitted trajectories of the neighbors $\Pi^{N^i}_{new}$, only the best plan in Π^i according to a metric is checked. If this plan fails the compatibility check, then the previous contingency is selected.
- At each step of the "while" loop in Algorithm 3.3.1 (lines 4-13), the implementation propagates an edge along a tree of trajectories using a sampling-based planner, instead of generating an entire trajectory. If the edge intersects t^i_{n+1}, the contingency $\gamma(t^i_{n+1} : \infty)$ is extended from $x(t^i_{n+1})$. If the contingency is collision-free and compatible with the past messages of neighbors in $\Pi^{N^i}_{prev}$, state $x(t^i_{n+1})$ is proven safe. Otherwise, it is unsafe and no future expansion of an edge is allowed past $x(t^i_{n+1})$.
- The sampling-based expansion of the tree structure of trajectories is biased:
  - A potential field in the workspace is used to promote the expansion of the tree towards the goal, while still maintaining probabilistic completeness [25].
  - To increase the probability of compatible plans, the tree expansion is biased away from other vehicles. This has a significant effect in the algorithm's performance.

Different sampling-based planners can be easily adapted to be employed in the above framework [21, 3]. It is also possible to consider the application of navigation or potential functions [40] and lattice-based methods [41].

- In the actual implementation, there is no need to differentiate between $P^{N^i}_{prev}$ and $P^{N^i}_{new}$. Each robot maintains a buffer for messages for each neighbor. As new trajectories are transmitted, they replace the part of old trajectories that has already been executed on the buffer. The implementation emphasizes the constraint that the robots have no access to a global clock and they do not know when each state along a transmitted trajectory is going to be executed.
- It is important to note, that even in the simulation environment employed for this work, it is difficult to perfectly synchronize the operation of all robots. This shows the importance of considering unsynchronized motion coordination algorithms.
- On the other hand, the latency in the experimental setup was relatively low. Thus, the situation described of Figure 3.2 did not arise. For this reason, the acknowledgement step was not included in the version of the algorithm used for this thesis' experiments, reducing in this way the number of peer-to-peer messages. For a version that can handle high latency with all the implementation details also taken care of as in Algorithm A.1.1, see Algorithm A.2.1.
- Robots all had a copy of the static environment map, so sensing was not explicitly modeled. All information on neighboring robots was thus delivered explicitly through communication.

4.2 Evaluation of Safety

To verify that the system implemented truly provides the guarantees presented in this thesis, three different cases were considered for the algorithm: (i) an implementation without contingencies, (ii) with contingencies but for robots with synchronized cycles and (iii) with contingencies and robots that are *not* synchronized. For each type of experiment the following figure reports the percentage of successful experiments. 20 experiments were executed for each case, averaging across synchronous and unsynchronized cases. The results presented clearly indicate that enabling contingencies results in a safe system in all cases.

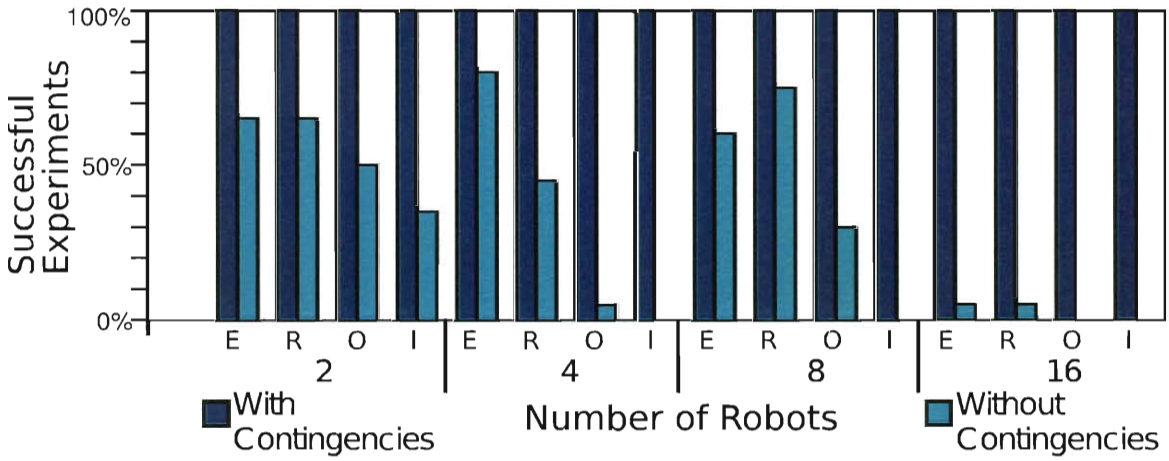


Figure 4.4: Graph showing 100% success rate with contingencies, and collision rate without for varying scenes and numbers of robots

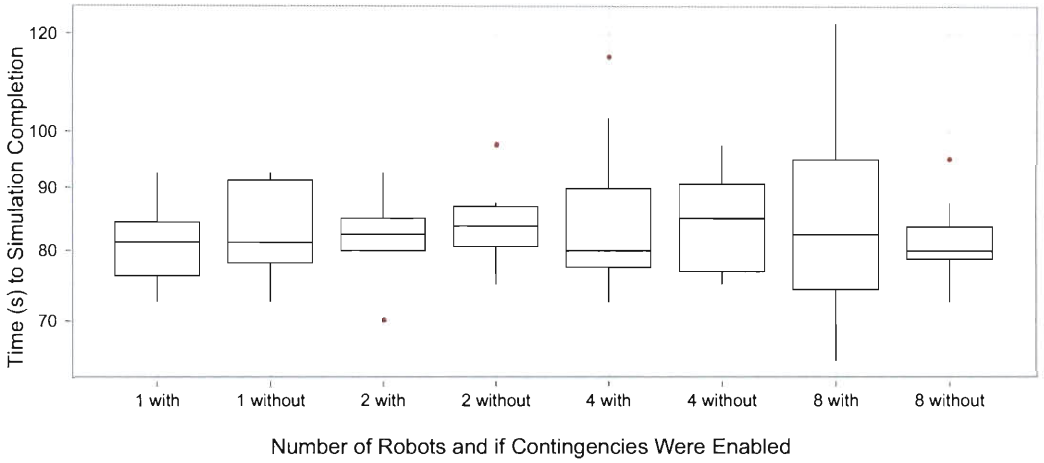


Figure 4.5: Performance difference caused by contingencies in the empty environment

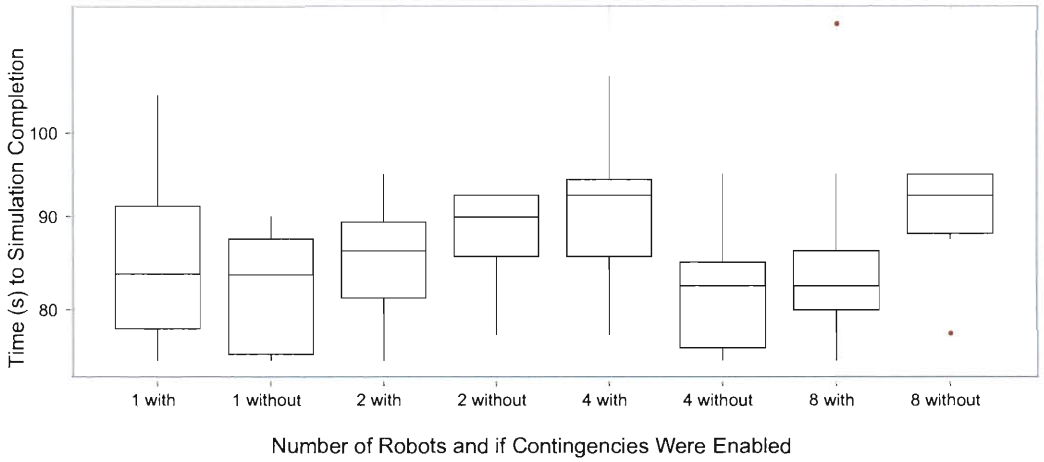


Figure 4.6: Performance difference caused by contingencies in the random environment

4.3 Scalability and Efficiency

Once the safety of the approach was confirmed, the focus turned on evaluating the effects of contingencies. A high-selection rate of contingencies is expected to decrease the performance of the robots, as these plans are not selected to make progress towards the

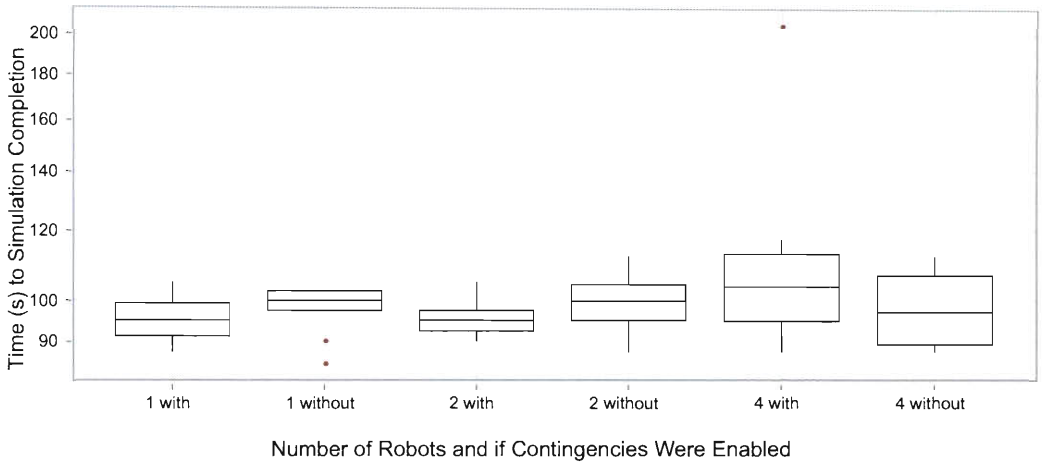


Figure 4.7: Performance difference caused by contingencies in the office environment

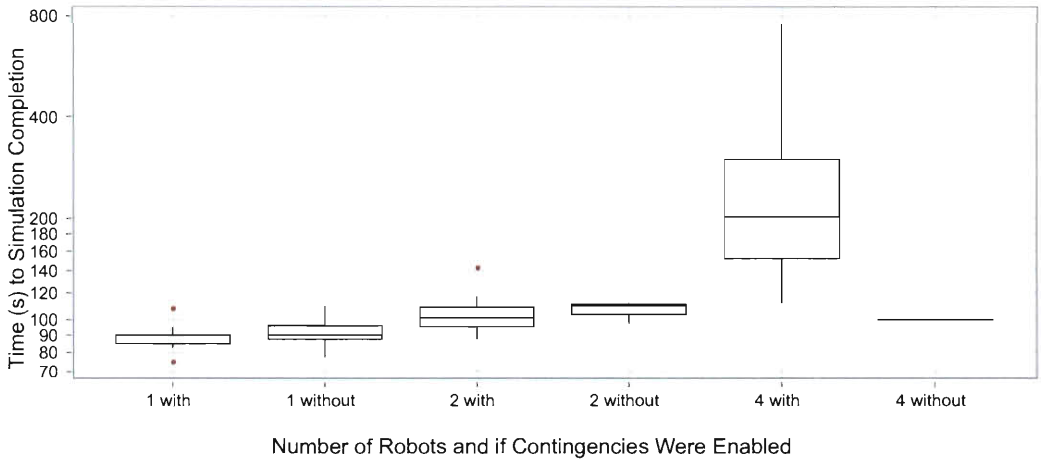


Figure 4.8: Performance difference caused by contingencies in the intersection environment

goal. The following table presents the average duration of experiments in seconds and the average velocity achieved by the robots both for the case without contingencies and the case with contingencies (both for synchronized and unsynchronized robots). The performance data without contingencies is from the cases where none of the robots entered ICS, which

means they often correspond to fewer than 20 experiments, and in some cases there is no successful experiment without contingencies to compare against.

For the cases that it is possible to compare against the solution without contingencies, it becomes apparent that the behavior of the robots is indeed more conservative and it takes more time to complete an experiment.

Although the algorithm has no progress guarantees, the randomized nature of the probabilistically complete planning algorithms helped to offset this. In practice, the simulations always eventually found a solution in the tested problems. The robots temporarily enter oscillatory motions, which were always eventually resolved. The initial set of experiments performed, however, while also successful, took approximately 10 times longer than the results in the above table. A local penalty for trajectories that brought an agent in close proximity to neighboring robots helped to reduce the occurrence of oscillatory motions - see Algorithm A.1.2 for details. Since these oscillations were the primary cause for long solution times, this small change resulted in a significant improvement in performance. The experiments presented here include this local penalty.

4.4 Synchronized versus Unsynchronized

Another objective of the experimentation procedure was to evaluate the differences in the performance of the algorithm between the synchronous and the unsynchronized case. In the synchronous case, all robots have a zero time offset but they are not aware of their synchronicity and they are not taking advantage of it as in previous work [2]. In the unsynchronized case, the offsets are the same across 10 averaged runs. These offsets are randomly

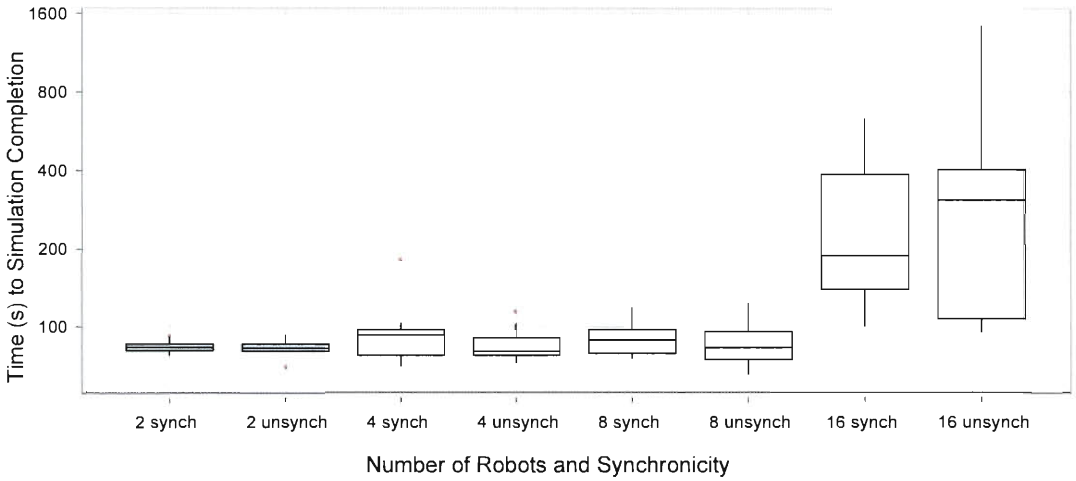


Figure 4.9: Performance difference caused by synchronicity in the empty environment

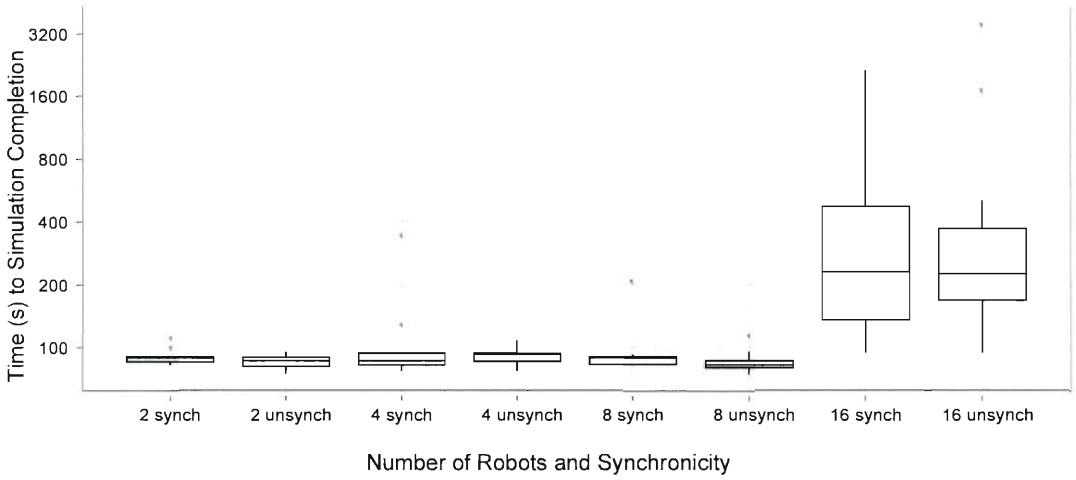


Figure 4.10: Performance difference caused by synchronicity in the random environment

precomputed and range from 0 to a maximum of $\frac{3}{4}$ of the planning cycle. Keeping the offsets well below a full planning cycle helped the simulation server process messages and responses correctly. Large offsets caused software instability but not a lack of safety in the protocol.

When the robots' cycles are synchronized, then it will be often the case that robots

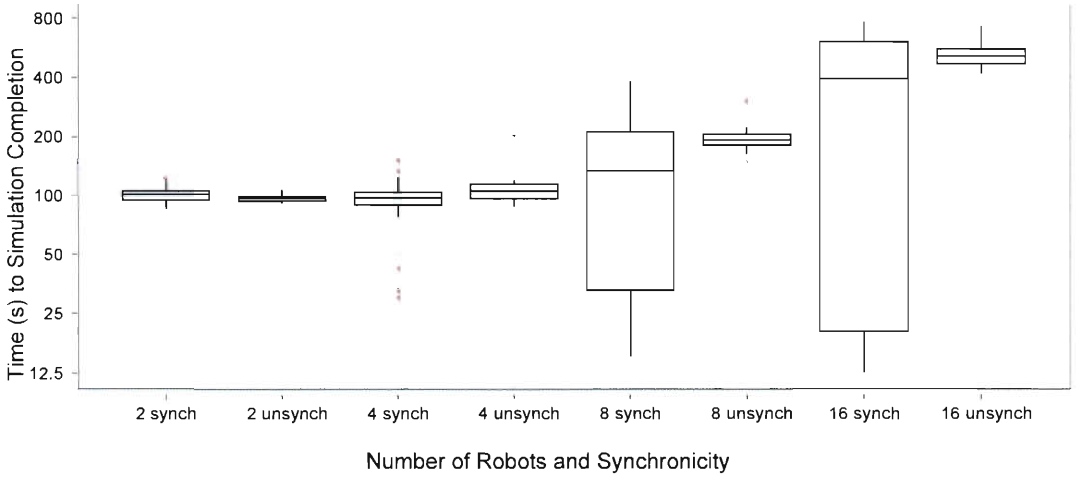


Figure 4.11: Performance difference caused by synchronicity in the office environment

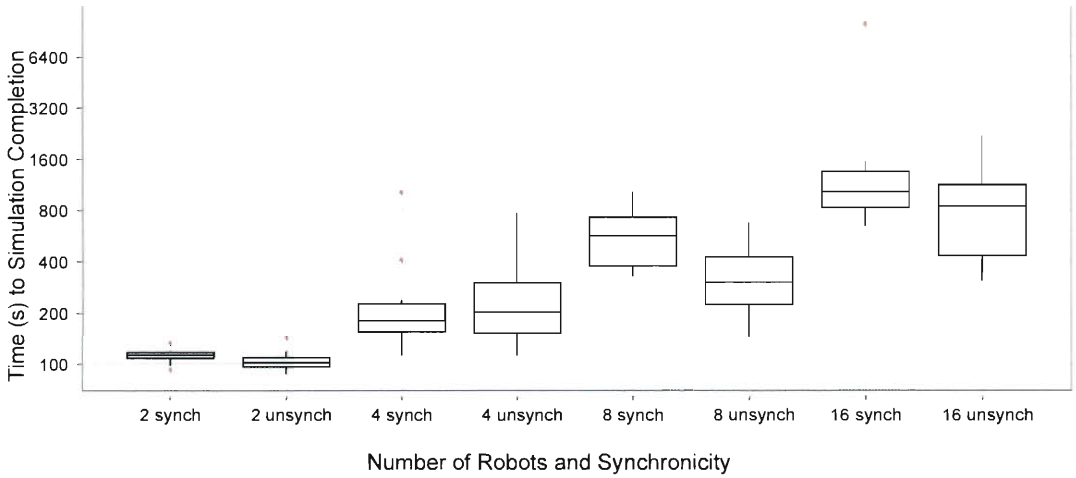


Figure 4.12: Performance difference caused by synchronicity in the intersection environment

are transmitting simultaneously, and potentially during the compatibility check of their neighbors. In certain cases this results in slightly longer durations for the completion of an experiment, as well as lower average velocities, but overall there is no consistent effect as in the random and empty scenes, there is a performance boost under synchronous operation, especially as the number of robots increases. In comparison to previous work

[2] where synchronicity was specifically taken advantage of, it is clear that the quality of the paths selected are lower in the current unsynchronized implementation. However, it is expected that further research in unsynchronized coordination algorithms can reduce this performance gap.

4.5 Scaling

Larger scale simulations for 32 and 48 robots were run to study the algorithm's scalability. For these cases, the approach without contingencies always fails. Note that as mentioned earlier, these robots have reduced sizes to reduce changes in completion time due to a cluttered environment.

Achieving safe unsynchronized operation for 48 second-order systems with the proposed setup is a significant achievement. The model of the agent employed is complex and the safety guarantees address the ICS issue. Furthermore, the simulation environment mimics the constraints of real-world communication between robots by running each agent on a separate processor and allowing only message-passing communication (TCP sockets). An experiment with 48 robots requires the employment of 49 processors (1 processor is used as a simulation server).

Parameter Evaluation An important parameter for the proposed approach is the duration of the planning cycle. For shorter durations of cycles, there was a higher deviation between runs and it was not possible to execute the larger experiments with 32 and 48 robots for a cycle duration less than 2 seconds. This limitation is due to the single thread running the world simulation. Cycle times of less than 1 second started showing problems with the

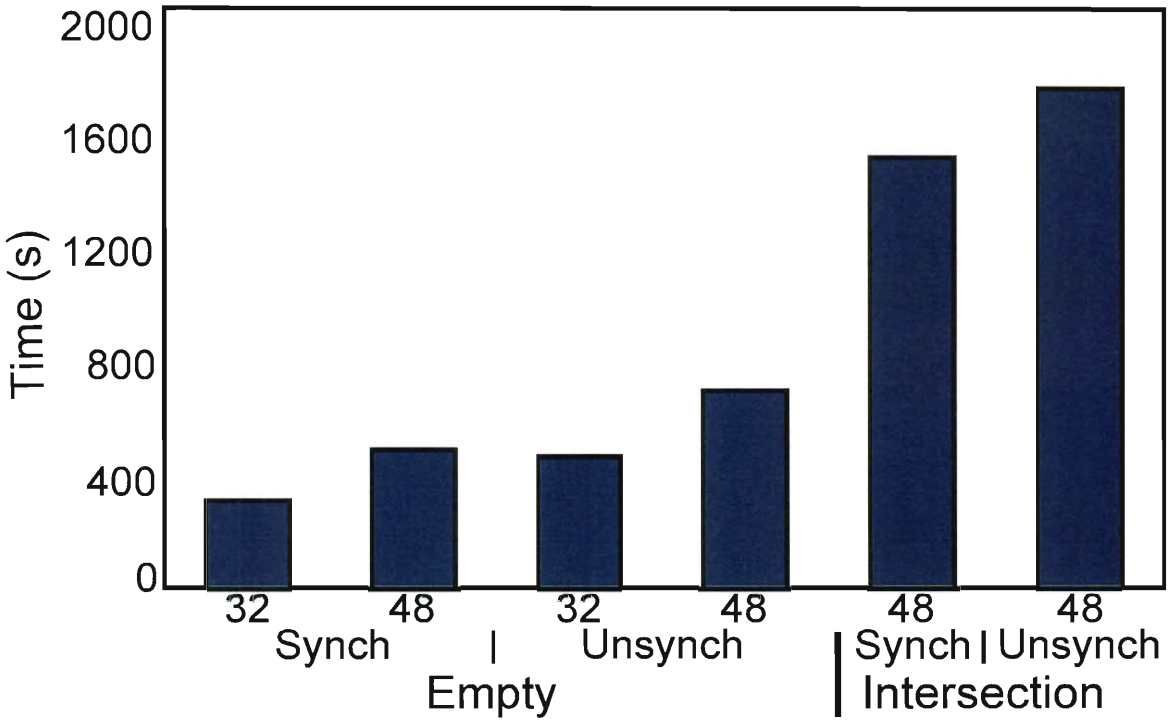


Figure 4.13: Graphs of the performance under 32 and 48 robot experiments, measured in time to completion (lower is better).

world server at even 16 robots, and are therefore not shown here. It is expected that the limit in hardware implementation would be dependent on the communication latency. The average completion time shows a noticeable increase as the duration of a cycle increases. The experiments presented in the previous tables were executed for a cycle duration of 2.5 seconds.

Planning Cycle		Number of Robots							
		2		4		8		16	
Scene	Cycle	Time	Vel.	Time	Vel.	Time	Vel.	Time	Vel.
Empty	1.0s	53.3	10.8	52.5	7.8	59.2	5.8	96.9	3.5
	1.5s	59.3	9.7	63.8	6.4	60.0	5.3	197.1	2.0
	2.0s	71.4	8.0	74.0	5.8	75.6	4.2	116.8	2.7
	2.5s	79.5	7.2	82.8	5.2	86.5	3.7	134.0	2.2
	3.0s	98.4	5.8	98.4	4.4	99.9	3.2	135.0	2.0
	3.5s	167.7	3.8	193.6	2.5	125.5	1.7	482.7	0.7

Table 4.1: The change in overall performance from varying cycle times

4.6 Periodic Contingencies for Systems with Minimum Velocity

A version of the car-like system that is required to have a positive minimum velocity was tested. This resembles the behavior of some aerial vehicles. For this system, the braking maneuver contingency is not valid, because the robots are not allowed to come to a halt. Instead, the contingencies employed require from the system to turn into the tightest circle possible without exceeding the limits on velocity and turning rate. Entering this contingency is more difficult, as it can require up to three separate controls:

1. Decelerate and turn to increase $|\zeta|$.

   In most cases, deceleration at maximum speed and steering at maximum rate will not hit their respective limits at the same time, therefore,

2. Decelerate or turn as above, until the component not at a limit value hits the limit.

3. Apply a zero control until a complete loop is made.

By integrating this maneuver for a certain duration, a complete periodic trajectory is defined, which can be used for collision checking and reason about safety over an infinite time horizon. This duration, $\tau_{periodic}$, can be upper-bounded given the other system parameters, similar to w_{max}:

$$\tau_{periodic} = \frac{(2 \cdot \pi)}{\zeta_{max}} + \frac{\zeta_{max}}{\phi_{max}} + \frac{w_{max}}{\alpha_{max}} \tag{4.1}$$

These three terms correspond to the time to

1. Traverse a full circle ($2 * \pi$ radians)
2. Turn to maximum turning
3. Decelerate to minimum velocity

Specifying a guaranteed safe start and goal state is significantly harder in environments with obstacles. In addition to tests in the previous environments shown, we have also run tests with a pattern of equilateral triangles as obstacles, as in Figure 4.14. This allowed us enough free space regions to more easily place robots safely, while still providing a challenging problem. Across 1000 simulations in this configuration with replanning cycle of 3.0s, we had a 100% success rate. When contingencies were disabled, the success rate dropped to 0% due to inter-robot collisions.

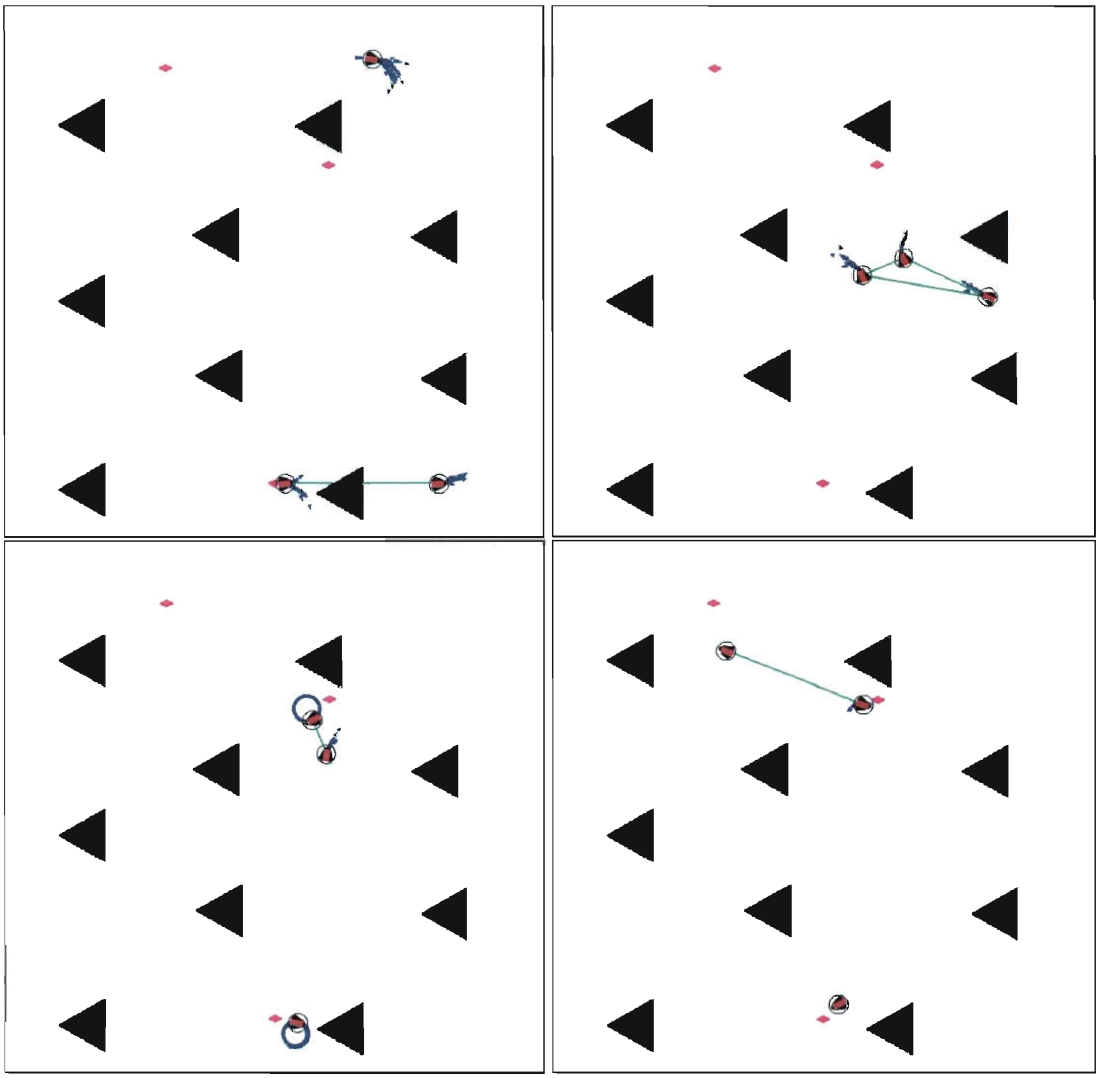


Figure 4.14: Geometric test environment for plane-like robots, showing 3 robots from start (upper left) to end (lower right). The circular plans clearly show the contingencies being executed.

Chapter 5

Discussion

This thesis presented a fully distributed and decentralized algorithm that guarantees ICS safety for a number of second-order robots that move purposely in the same environment. A proof of safety for this protocol is provided. Simulations confirm that the framework indeed provides safety, is scalable and easily adapted to novel systems.

In the future, we would like to investigate making additional guarantees about performance. It possible that the scheme could be proven complete if there were a system in place to promote a vehicle to be a leader. After promotion, it would be given priority over all other vehicles. All other robots would be required to respect this vehicle, which, using adaptive time stepping [49] could be locally complete. However, proving completeness for this combined scheme appears significantly more complex than the case of dynamic networks, above. This approach would not have the same problem with scalability, however, and may provide benefits even if it could not be used to prove completeness.

The approach developed in this thesis could also be extended to the framework of dy-

namic networks [57], where robots that define a connected component in the communication graph define a composite robot, for which planning takes place in a centralized manner. This may allow the consideration of even more complex scenarios. Although this may allow for completeness guarantees in terms of multi-robot planning for systems with second-order dynamics (although the introduction of sampling-based planners weaken these guarantees), scalability can be an issue in any work along this direction.

Additional work to expand on the protocol presented in this thesis could include:

- considering robots with different durations for planning cycles,
- dealing with unreliable communication,
- distributed optimization for improving the quality of paths selected despite the unsynchronized operation,
- addressing tasks that go beyond moving from initial to final states,
- comparing directly to reactive methods experimentally,
- construct a better world simulation that could handle more robots,
- investigating the communication costs more explicitly, and investigate ways to either reduce the amount of communication, or trade it off for additional computation.

We conclude that planning with communication is a very powerful mode of operation. Its primary drawback is that it tends to be expensive, both in terms of computation and communication. The benefit of this expense, however, is the safety guarantees that can be made in a very general framework. Exploring a trade-off between communication, local

computation, and conservatism with respect to assumptions of robots' future behavior is a very interesting line of research, while staying in the same general mode of operation of planning with communication. The ease of adding a new vehicle model with radically different contingency plan requirements due to its new dynamics shows off the strengths of this novel coordination scheme. As robotics hardware becomes cheap ubiquitous, it is anticipated that the cost of a heavy-weight protocol will be more than offset by the ease of implementation, particularly when strong guarantees can be made.

Bibliography

[1] J. Reif and H. U. C. M. A. C. LAB., *Complexity of the generalized mover's problem.* Norwood, NJ: Ablex Publishing Corporation, 1985, pp. 267–281. [Online]. Available: http://www.cs.duke.edu/~reif/paper/movers.pdf

[2] K. E. Bekris, K. Tsianos, and L. E. Kavraki, "Safe and Distributed Kinodynamic Replanning for Vehicular Networks," *Mobile Networks and Applications*, vol. 14, no. 3, pp. 292–308, 2009.

[3] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized Kinodynamic Motion Planning with Moving Obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, Mar. 2002.

[4] S. Petti and T. Fraichard, "Safe Motion Planning in Dynamic Environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, AB, Canada, Aug. 2005.

[5] O. Brock and O. Khatib, "High-speed Navigation Using the Global Dynamic Window Approach," in *IEEE ICRA*, Detroit, MI, USA, May 1999.

[6] M.-I. Jung and J.-H. Kim, "Development of a Fault-Tolerant Omnidirectional Wheeled Mobile Robot Using Nonholonomic Constraints," *The International Journal of Robotics Research*, vol. 21, no. 5-6, pp. 527–539, May 2002. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/027836402761393379

[7] S. Loizou and K. Kyriakopoulos, "Closed loop navigation for multiple holonomic vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, no. October. IEEE, 2002, pp. 2861–2866. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1041704

[8] R. Murray and S. Sastry, "Nonholonomic motion planning: steering using sinusoids," *IEEE Transactions on Automatic Control*,

vol. 38, no. 5, pp. 700–716, May 1993. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=277235

[9] G. Oriolo and Y. Nakamura, "Free-joint manipulators: motion control under second-order nonholonomic constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, no. 91. IEEE, 1991, pp. 1248–1253. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=174671

[10] ——, "Control of mechanical systems with second-order nonholonomic constraints: Underactuated manipulators," in *IEEE Conference on Decision and Control*. IEEE, 2002, pp. 2398–2403. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=261620

[11] L. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957. [Online]. Available: http://www.jstor.org/stable/2372560

[12] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990. [Online]. Available: http://www-personal.acfr.usyd.edu.au/spns/motion/ReedsShepp1990.pdf

[13] J. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," *Robot motion planning and control*, pp. 1–53, 1998. [Online]. Available: http://www.springerlink.com/index/u6nw7085w2q3l06l.pdf

[14] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," *Robot motion planning and control*, pp. 171–253, 1998. [Online]. Available: http://www.springerlink.com/index/l14472m6jm001607.pdf

[15] C. Samson, "Velocity and torque feedback control of a nonholonomic cart," *Advanced robot control*, pp. 125–151, 1991. [Online]. Available: http://www.springerlink.com/index/c16q383884661073.pdf

[16] a. De Luca and G. Oriolo, "Local incremental planning for nonholonomic mobile robots," in *IEEE International Conference on Robotics and Automation*. IEEE Comput. Soc. Press, 1994, pp. 104–110. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=351003

[17] K. M. Lynch, "Collision-Free Trajectory Planning for a 3-DoF Robot with a Passive Joint," *The International Journal of Robotics Research*, vol. 19, no. 12, pp. 1171–1184, Dec. 2000. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/02783640022068011

[18] S. M. LaValle, "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/02783640122067453

[19] R. Brennan, "Performance comparison and analysis of reactive and planning-based control architectures for manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 16, no. 2-3, pp. 191–200, Apr. 2000. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0736584500000077

[20] J. Reif and M. Sharir, "Motion Planning in the Presence of Moving Obstacles," in *IEEE International Symposium on Foundations of Computer Science*, Portland, OR, 1985.

[21] S. M. LaValle and J. J. Kuffner, "Randomized Kinodynamic Planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[22] T. Fraichard and H. Asama, "Inevitable Collision States: A Step Towards Safer Robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004. [Online]. Available: http://emotion.inrialpes.fr/bibemotion/2004/FA04/

[23] L. Martinez-Gomez and T. Fraichard, "Collision Avoidance in Dynamic Environments: An ICS-based solution and its Comparative Evaluation," in *IEEE International Conference on Robotics and Automation*, Kobe, Japan, May 2009.

[24] T. Fraichard, "A Short Paper about Motion Safety," in *IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007.

[25] K. E. Bekris and L. E. Kavraki, "Greedy but Safe Replanning under Kinodynamic Constraints," in *IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 2007.

[26] J. Borenstein and Y. Korem, "The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, 1991.

[27] O. Brock and O. Khatib, "Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths," in *IEEE International Conference on Robotics and Automation.* IEEE, 2000, pp. 550–555. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=844111

[28] ——, "Elastic Strips: Real-Time Path Modification for Mobile Manipulation," in *International Symposium of Robotics Research*, 1997, pp. 5–13. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.5151

[29] F. Lamiraux, D. Bonnafous, and O. Lefebvre, "Reactive path deformation for nonholonomic mobile robots," in *IEEE Transactions on Robotics*, vol. 20, no. 6, 2004, pp. 967–977.

[30] Y. Yang and O. Brock, "Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments," in *Robotics: Science and Systems*, 2006.

[31] J. Minguez and L. Montano, "Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 45–59, 2004.

[32] D. Fox, W. Burgard, and S. Thrun, "The Dynamic Window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, 1997.

[33] M. Seder and I. Petrovic, "Dynamic Window based Approach to Mobile Robot Motion Control in the Presence of Moving Obstacles," in *IEEE International Conference on Robotics and Automation*, Apr. 2007.

[34] P. Fiorini and Z. Shiller, "Motion Planning in Dynamic Environments Using Velocity Obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, 1998.

[35] F. Large, C. Laugier, and Z. Shiller, "Navigation Among Moving Obstacles Using the {NLVO}: Principles and Applications to Intelligent Vehicles," *Autonomous Robots*, vol. 19, no. 2, 2005.

[36] J. den Berg, M. Lin, and D. Manocha, "Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation," in *IEEE International Conference on Robotics and Automation*, 2008.

[37] J. van den Berg, J. Snape, S. Guy, and D. Manocha, "Reciprocal Collision Avoidance with Acceleration-Velocity Obstacles," in *IEEE International*

Conference on Robotics and Automation, Shanghai, 2011. [Online]. Available: http://gamma.cs.unc.edu/AVO/publications/AVO.pdf

[38] E. Lalish and K. A. Morgansen, "Decentralized Reactive Collision Avoidance for Multivehicle Systems," in *IEEE Conference on Decision and Control*, 2008.

[39] L. Pallottino, V. G. Scordio, A. Bicchi, and E. Frazzoli, "Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1170–1183, Dec. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4392815

[40] D. V. Dimarogonas and K. J. Kyriakopoulos, "Decentralized Navigation Functions for Multiple Robotic Agents with Limited Sensing Capabilities," *Journal of Intelligent and Robotic Systems*, vol. 48, no. 3, pp. 411–433, Jan. 2007. [Online]. Available: http://www.springerlink.com/index/10.1007/s10846-006-9113-x

[41] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially Constrained Mobile Robot Motion Planning in State Lattices," in *Journal of Field Robotics*, vol. 26, no. 3, 2009, pp. 308–333.

[42] M. S. Wikman, M. Branicky, and W. S. Newman, "Reflexive Collision Avoidance: A Generalized Approach," in *IEEE International Conference on Robotics and Automation*, 1993.

[43] J. Bruce and M. Veloso, "Real-Time Multi-Robot Motion Planning with Safe Dynamics," in *International Workshop on Multi-Robot Systems*, A. Schultz, L. Parker, and F. Schneider, Eds., 2003.

[44] E. Frazzoli, M. Dahleh, and E. Feron, "Real-Time Motion Planning for Agile Autonomous Vehicles," *AIAA Journal of Guidance, Control and Dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[45] M. Kalisiak and M. de Panne, "Faster Motion Planning using Learned Local Viability Models," in *IEEE International Conference on Robotics and Automation*, Roma, Italy, May 2007.

[46] N. Chan, J. J. Kuffner, and M. Zucker, "Improved Motion Planning Speed and Safety using Regions of Inevitable Collision," in *CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, Jul. 2008.

[47] R. Alami, T. Simeon, and K. M. Krishna, "On the Influence of Sensor Capacities and Environment Dynamics Onto Collision-Free Motion Plans," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Lausanne, CH, 2002.

[48] R. Vatcha and J. Xiao, "Perceived {CT-Space} for Motion Planning in Unknown and Unpredictable Environments," in *Workshop on the Algorithmic Foundations of Robotics*, Mexico, 2008.

[49] K. Hauser, "Adaptive Time Stepping in Real-Time Motion Planning," in *Workshop on the Algorithmic Foundations of Robotcs*. Springer, 2011, pp. 139–155. [Online]. Available: http://www.springerlink.com/index/K1350J4820K1M148.pdf

[50] L. Chaimowicz, T. Sugar, V. Kumar, and M. Campos, "An architecture for tightly coupled multi-robot cooperation," in *IEEE International Conference on Robotics and Automation*, vol. 3. IEEE, 2005, pp. 2992–2997. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=933076

[51] D. Goldberg, V. Cicirello, M. Dias, R. Simmons, S. Smith, and A. Stentz, "Market-based multi-robot planning in a distributed layered architecture," in *International Workshop on Multi-Robot Systems*, vol. 2, 2003, pp. 27–38.

[52] G. Pereira, A. Das, V. Kumar, and M. Campos, "Decentralized motion planning for multiple robots subject to sensing and communication constraints," in *International Workshop on Multi-Robot Systems*. Springer Netherlands, 2003, p. 267.

[53] K. Azarm and G. Schmidt, "Conflict-free motion of multiple mobile robots based on decentralized motion planning and negotiation," in *IEEE International Conference on Robotics and Automation*, no. April. IEEE, 1997, pp. 3526–3533. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=606881

[54] Y. Cao, A. Fukunaga, A. Kahng, and F. Meng, "Cooperative mobile robotics: antecedents and directions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 27. IEEE Comput. Soc. Press, 1997, pp. 226–234. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=525801

[55] S. Chien, A. Barrett, T. Estlin, and G. Rabideau, "A comparison of coordinated planning methods for cooperating rovers," in *International Conference on Autonomous Agents*. New York, New York, USA: ACM Press, 2000, pp. 100–101. [Online]. Available: http://portal.acm.org/citation.cfm?doid=336595.337057

[56] G. Sanchez and J.-C. Latombe, "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems," in *IEEE International Conference on Robotics and Automation*, no. May. IEEE, 2002, pp. 2112–2119. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1014852

[57] C. Clark, S. Rock, and J.-C. Latombe, "Motion Planning for Multi-Robot Systems using Dynamic Robot Networks," in *IEEE International Conference on Robotics and Automation*, Taipei, Taiwan, May 2003.

[58] M. Erdmann and T. Lozano-Perez, "On Multiple Moving Objects," in *IEEE International Conference on Robotics and Automation*, 1986, pp. 1419–1424.

[59] J. Peng and S. Akella, "Coordinating Multiple Robots with Kinodynamic Constraints Along Specified Paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005.

[60] V. J. Lumelsky and K. R. Harinarayan, "Decentralized motion planning for multiple mobile robots: The cocktail party model," *Autonomous Robots*, vol. vol, pp. 4pp121—-135, 1997.

[61] L. Parker, "Cooperative Robotics for Multi-Target Observation," *Intelligent Automation and Soft Computing*, vol. 5, pp. 5–19, 1999. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.5511

[62] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, "Impact of Interference on Multi-Hop Wireless Network Performance," *Wireless Networks*, vol. 11, no. 4, pp. 471–487, Jul. 2005. [Online]. Available: http://www.springerlink.com/index/10.1007/s11276-005-1769-9

[63] Y. Yi and S. Shakkottai, "Hop-by-Hop Congestion Control Over a Wireless Multi-Hop Network," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 133–144, Feb. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4100728

[64] R. Draves, J. Padhye, and B. Zill, *Routing in multi-radio, multi-hop wireless mesh networks*. New York, New York, USA: ACM Press, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1023720.1023732

[65] M. R. Benjamin, *Multi-objective autonomous vehicle navigation in the presence of cooperative and adversarial moving contacts*. IEEE, 1878. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1191917

[66] N. Agmon, S. Kraus, and G. a. Kaminka, "Multi-robot perimeter patrol in adversarial settings," in *IEEE International Conference on Robotics and Automation*. IEEE, May 2008, pp. 2339–2345. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543563

[67] B. Browning, J. Bruce, M. Bowling, and M. Veloso, "STP: skills, tactics, and plays for multi-robot control in adversarial environments," *IEEE Journal of Control and Systems Engineering*, vol. 219, no. 1, pp. 33–52, Jan. 2005. [Online]. Available: http://http://dx.doi.org/10.1243/095965105X9470

[68] R. Beard and E. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference*. IEEE, 2005, pp. 1859–1864. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1470239

[69] J.-P. Laumond, Ed., *Robot Motion Planning and Control*, ser. Lectures Notes in Control and Information Sciences 229. Springer, 1998.

Appendix A

Algorithms

A.1 Exact Algorithms Used in Simulation

Algorithm A.1.1 Safe and unsynchronized algorithm, as actually implemented

1: $\Pi^i \leftarrow \emptyset$, $\Pi^{N^i}_{prev} \leftarrow \emptyset$, $\Pi^{N^i}_{new} \leftarrow \emptyset$
2: **for all** $R^j \in N^i$ **do**
3: $\Pi^{N^i}_{prev} \leftarrow \Pi^{N^i}_{prev} \cup \pi^j(\, \pi^j(x^j(t^j_{n-1}), p^j(t^j_{n-1} : t^j_n)),\ \gamma(t^j_n : \infty)\,)$
(i.e., include all past trajectories and attached contingencies of neighbors)
4: **end for**
5: **for all** $R^j \in N^i$ **do**
6: Find the most recent current state information of R^j
7: Find the first instance of this state in $\Pi^{N^i}_{prev}$.
8: Remove all states in $\Pi^{N^i}_{prev}$ that were transmitted before the first coincidence between current and previously-transmitted states {This assumes states arrive in order of execution, but does not assume that a plan can be differentiated from a contingency}
9: **end for**
10: **while** $t < t^i_n - \epsilon$ **do**
11: $\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})) \leftarrow$ collision-free trajectory from a single-robot planner
12: $\pi^i_\gamma \leftarrow \pi^i(\, \pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})),\ \gamma(t^i_{n+1} : \infty))$ (i.e., contingency concatenation)
13: **if** $\forall\, t \in [t^i_{n+1} : \infty) :\ x[\pi^i_\gamma](t) \in \mathcal{X}_f$ **then**
14: $\Pi^i \leftarrow \Pi^i \cup \pi^i_\gamma$
15: **for all** $\pi^j_\gamma \in \Pi^{N^i}_{prev}$ **do**
16: **if** $\pi^i_\gamma \not\asymp \pi^j_\gamma$ **then**
17: $\Pi^i \leftarrow \Pi^i - \pi^i_\gamma$
18: **end if**

19: **end for**
20: **end if**
21: **if** $R^j \in N^i$ is transmitting a trajectory and an attached contingency **then**
22: $\Pi^{N^i}_{new} \leftarrow \Pi^{N^i}_{new} \cup \pi^j_\gamma(\, \pi^j(x^j(t^j_n), p^j(t^j_n : t^j_{n+1})),\ \gamma(t^j_{n+1} : \infty)\,)$
23: **end if**
24: **end while**
25: **for all** $\pi^i_\gamma \in \Pi^i$ **do**
26: **for all** $\pi^j_\gamma \in \Pi^{N^i}_{new}$ **do**
27: **if** $\pi^i_\gamma \not\asymp \pi^j_\gamma$ **then**
28: $\Pi^i \leftarrow \Pi^i - \pi^i_\gamma$
29: **end if**
30: **end for**
31: **end for**
32: **if** Π^i empty or if a message was received during compatibility check **then**
33: $\pi^i_* \leftarrow \pi^i(x^i(t^i_n), \gamma(t^i_n : \infty))$ (i.e., follow the available contingency for next cycle)
34: **else**
35: Run Algorithm A.1.2 to generate a potential map
36: $\pi^i_* \leftarrow$ the safe trajectory concatenation in Π^i which has the best total payoff based on the potential map, evaluated with an exponential decay as planning tree depth increases
37: **end if**
38: **Transmit** π^i_* to all neighbors in N^i {$\epsilon_{collisioncheck}$ is the maximum time it will take to run collision checking between two plans}
39: **while** $t < t^i_n - \epsilon_{collisioncheck}$ **do**
40: **if** A message was received from R^j after transmission **then**
41: $\Pi^{N^i}_{new} \leftarrow \Pi^{N^i}_{new} \cup \pi^j_\gamma(\, \pi^j(x^j(t^j_n), p^j(t^j_n : t^j_{n+1})),\ \gamma(t^j_{n+1} : \infty)\,)$
42: **if** $pi^i_* \not\asymp \pi^j_*$ **then**
43: $\pi^i_* \leftarrow \gamma(t^i_n : \infty)$ {If the plan we have already transmitted to our neighbors is in conflict with the message we received, revert to contingency immediately.}
44: **end if**
45: **end if**
46: **end while**
47: **Execute** π^i_* during next cycle

Algorithm A.1.2 Potential Map Creation and Plan Selection

$DistanceMap := Map(-1.0)$
$DistanceMap := D \cup Pair(Goal, 0.0)$
while $\neg Complete(DistanceMap)$ **do**
  for all $\{C : C \in Map \wedge C \notin Obstacles$ **do**
    $MinD = 0.0$
    for all $\{C : C$ adjacent $\{d : d \in DistanceMap\}\}$ **do**
      $MinD := Min(MinD, d_{dist} + DistWeight)$
    end for
    $D := D \cup Pair(C, MinD$
  end for
end while
$PotentialMap := DistanceMap$
for all $T \in \{0, ReplanningCycle\}$ **do**
  for all $\{R : Distance(StateAtTime(R, T), Self) < CommRange\}$ **do**
    for all $\{C : C \in Map \wedge C \notin Obstacles \wedge Distance(C, StateAtTime(R, T) < FALLOFF\}$ **do**
      $PotentialMap[C]_{weight} += RobotWeight$
    end for
  end for
end for
for all $x_t^i \in \Pi_{new}^{N^i} \cup \Pi_{prev}^{N^i}$ **do**
  for all $d \in [\mathbb{S}...2 \cdot \mathbb{S}]$ **do**
    $ExtraCostArea \leftarrow PotentialMap$ at distance d from state x_t^i
    $ExtraCostArea += \frac{\mathbb{A}}{d}$ {$\mathbb{A}$ is an arbitrary extra cost amount to add. This implementation used $\frac{1}{10}$ of the Cartesian extent of the workspace.}
  end for
end for
$BestPlan := PlanningTree_{plans}[0]$
for all $\{P : P \in PlanningTree_{plans}\}$ **do**
  if $PotentialMap[P]_{weight} < BestPlan_{weight}$ **then**
    $BestPlan := P$
  end if
end for

A.2 Safe Algorithm Under High Latency

Algorithm A.2.1 Safe and unsynchronized algorithm including acknowledgement messages

$\Pi^i \leftarrow \emptyset$, $\Pi^{N^i}_{prev} \leftarrow \emptyset$, $\Pi^{N^i}_{new} \leftarrow \emptyset$
for all $R^j \in N^i$ **do**
  $\Pi^{N^i}_{prev} \leftarrow \Pi^{N^i}_{prev} \cup \pi^j(\, \pi^j(x^j(t^j_{n-1}), p^j(t^j_{n-1} : t^j_n)),\ \gamma(t^j_n : \infty)\,)$
  (i.e., include all past trajectories and attached contingencies of neighbors)
end for
for all $R^j \in N^i$ **do**
  Find the most recent current state information of R^j
  Find the first instance of this state in $\Pi^{N^i}_{prev}$.
  Remove all states in $\Pi^{N^i}_{prev}$ that were transmitted before the first coincidence between current and previously-transmitted states {This assumes states arrive in order of execution, but does not assume that a plan can be differentiated from a contingency}
end for
while $t < t^i_n - \epsilon$ **do**
  $\pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})) \leftarrow$ collision-free trajectory from a single-robot planner
  $\pi^i_\gamma \leftarrow \pi^i(\, \pi^i(x^i(t^i_n), p^i(t^i_n : t^i_{n+1})),\ \gamma(t^i_{n+1} : \infty))$ (i.e., contingency concatenation)
  if $\forall\, t \in [t^i_{n+1} : \infty) :\ x[\pi^i_\gamma](t) \in \mathcal{X}_f$ **then**
    $\Pi^i \leftarrow \Pi^i \cup \pi^i_\gamma$
    for all $\pi^j_\gamma \in \Pi^{N^i}_{prev}$ **do**
      if $\pi^i_\gamma \not\asymp \pi^j_\gamma$ **then**
        $\Pi^i \leftarrow \Pi^i - \pi^i_\gamma$
      end if
    end for
  end if
  if $R^j \in N^i$ is transmitting a trajectory and an attached contingency, with header containing message ID number **then**
    $\Pi^{N^i}_{new} \leftarrow \Pi^{N^i}_{new} \cup \pi^j_\gamma(\, \pi^j(x^j(t^j_n), p^j(t^j_n : t^j_{n+1})),\ \gamma(t^j_{n+1} : \infty)\,)$
    Transmit an acknowledgement message to R^j containing the same message ID number
  end if
end while
for all $\pi^i_\gamma \in \Pi^i$ **do**
  for all $\pi^j_\gamma \in \Pi^{N^i}_{new}$ **do**
    if $\pi^i_\gamma \not\asymp \pi^j_\gamma$ **then**

$\Pi^i \leftarrow \Pi^i - \pi^i_\gamma$
end if
end for
end for
if Π^i empty **then**
$\pi^i_* \leftarrow \pi^i(x^i(t^i_n), \gamma(t^i_n : \infty))$ (i.e., follow the available contingency for next cycle)
else
Run Algorithm A.1.2 to generate a potential map
$\pi^i_* \leftarrow$ trajectory in Π^i which has the best total payoff based on the potential map, evaluated with an exponential decay as planning tree depth increases
end if
Transmit π^i_* to all neighbors in N^i, with header containing a unique message ID {$\epsilon_{collisioncheck}$ is the maximum time it will take to run collision checking between two plans}
while $t < t^i_n - \epsilon_{collisioncheck}$ **do**
if A message was received from R^j after transmission **then**
$\Pi^{N^i}_{new} \leftarrow \Pi^{N^i}_{new} \cup \pi^j_\gamma(\, \pi^j(x^j(t^j_n), p^j(t^j_n : t^j_{n+1})),\ \gamma(t^j_{n+1} : \infty)\,)$
if $pi^i_* \not\asymp \pi^j_*$ **then**
$\pi^i_* \leftarrow \gamma(t^i_n : \infty)$ {If the plan we have already transmitted to our neighbors is in conflict with the message we received, revert to contingency immediately.}
end if
end if
end while
if All expected acknowledgments have not been received **then**
$\pi^i_* \leftarrow \gamma(t^i_n : \infty)$ {Our neighbors have not accepted responsibility for avoiding our plan, revert to contingency}
end if
Execute π^i_* during next cycle